1

LEVEL II

AD-A062808

R-1136

A NEW BASELINE FOR THE INERTIAL NAVIGATION STRAPDOWN SIMULATOR PROGRAM

VOLUME II Analytical Development

by

R.J. Nuss, J.T. Prohaska, D.G. Riegsecker

July 1978

The Charles Stark Draper Laboratory, Inc.

Cambridge, Massachusetts 02139

78 [illegible] 023

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER R-1136	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A New Baseline for the Inertial Navigation Strapdown Simulator Volumes I, II, III, and IV		5. TYPE OF REPORT & PERIOD COVERED 6/1/77 - 7/15/78
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. Nurse, J. Prohaska, D. Riegsecker		8. CONTRACT OR GRANT NUMBER(s) F33615-75-C-1149
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Charles Stark Draper Laboratory, Inc. 555 Technology Sq., Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 6095, Task 4.2.5
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory Wright-Patterson AFB, Dayton, Ohio 45433		12. REPORT DATE July 1978
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

ACCESSION for		
NTIS	White Section	☒
DDC	Buff Section	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	AVAIL. and/or	SPECIAL
A		

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Inertial Navigation
Strapdown
Simulation
Instrument Modeling
Laser Gyros
Random Vibration
Single degree of freedom gyros and accelerometers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This four-volume report describes an updated and expanded version of a direct, digital, modular simulation of a strapdown inertial navigation system employing a wander-azimuth computational frame, and subject to a six degree of freedom random vibration environment. The original version of this simulation was developed under Task 4.2.3(a) of the above contract during 1975 and 1976.

(CONTINUED ON REVERSE)

The user may simulate not only the gross dynamics of the flight profile (from an external or internal profile generation) but also the angular and linear random vibrations resulting from gusts and turbulence acting on the airframe. The total environment is applied to the models of the inertial components (laser or SDR gyros and pendulous accelerometers). The resulting outputs of simulated IMU are summed in an interface module and compensated and scaled in the simulated navigation computer. The latter also contains the velocity/attitude algorithm, which computes the body-to-inertial transformation, using either the direction cosine matrix or quaternion, and the navigation algorithm which numerically integrates the specific forces after transformation to the local vertical, wander azimuth computational frame. The outputs of the simulated navigation computer are the computed position, velocity, and attitude of the vehicle with respect to a local vertical, north pointing frame. The flight profile and the differences between it and the simulated navigation computer outputs are tabulated in an evaluation module for printing, plotting, or post processing.

A ground alignment Kalman filter for the INSS, also developed under this task, is not documented in this report, but may be available from AFAL/RWA-2 or -3.

The program is written in Fortran IV for use on a CD6600/CYBER74.

The report is structured as follows:

- Volume I is the Introduction and Summary
- Volume II contains analytical development of the equations to be mechanized and the transition to difference equation form
- Volume III is the Program Description and User's Guide
- Volume IV contains Program Listings.

R-1136

A NEW BASELINE FOR THE INERTIAL NAVIGATION STRAPDOWN SIMULATOR PROGRAM

VOLUME II Analytical Development

by

R.J. Nurse, J.T. Prohaska, D.G. Riegsecker

July 1978

Approved: W Denhard

W. Denhard

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

ACKNOWLEDGEMENT

This four-volume report was prepared under USAF Contract No. F33615-75-C-1149 by the Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts, in accordance with Section 4 of the contract. The monitoring Air Force project engineer is Mr. Neil Banke (RWA-2), Air Force Avionics Laboratory, Dayton, Ohio.

The Draper Laboratory program manager for this task is John Harper and the lead engineer is Roy Nurse. The authors of the report are Roy Nurse, John Prohaska, and Darold Riegsecker.

The authors express their appreciation to Messrs. William Shephard, David Kaiser, Stanton Musick, and Maj. Robert Edwards of the Air Force Avionics Laboratory and William Ostanek, Jane Goode, Tom Thorvaldsen, and William Caffery of the C. S. Draper Laboratory for their assistance during the course of this contract.

VOLUME II

TABLE OF CONTENTS

VOLUME II

TABLE OF CONTENTS
(Cont'd)

VOLUME II

LIST OF ILLUSTRATIONS

VOLUME III

LIST OF TABLES

SECTION I

INTRODUCTION

1.1 Objectives and Rationale

This volume contains the analytical development of the equations required for mechanization of a strapdown inertial navigation system simulation program.

One objective of this volume is to help clarify the meanings of the relationships between the multiply-defined coordinate frames which occur throughout the program. These multiply-defined frames do not materially complicate the coding but can impede analytical insight unless fully understood. This effort is concentrated in the Appendices (mainly Appendix B) and the results are simply introduced in the main body.

Another objective is to provide the necessary analytical foundation for the equations which often merely appeared in earlier documentation. Program structure was unchanged but several modules were reworked: the laser gyro and compensation are completely new; the navigation and velocity-attitude algorithm are upgraded per (5).

The approach to providing the analytical development of the equations actually coded started with inverse Fortran transformation of the code - to obtain the difference equations mechanized - followed by rederivation of these equations and corrections where necessary.

1.2 Summary

Starting with discussion of the need filled by the INSS and the general structure of the program, the remainder of the volume derives, develops, and the$\overset{h}{\wedge}$ presents revelant algorithms, generally on a module-by-module basis.

Section 3 describes the interim trajectory module-a self-contained driver which does not permit aircraft maneuvers. Although, the predecessor of this module was used almost exclusively in the earlier version of program, it was not documented in the reports. This time the trajectory module, which interfaces with PROFGEN, is omitted since it and a modified PROFGEN program are currently available at AFAL.

Section 4 develops and presents the equations and algorithms necessary to simulate directly the random linear and angular motion of the vehicle defined by six displacement power spectral densities. The resultant random motion is superimposed on the trajectory output and passed on to the inertial component modules but not to the evaluation module, hence vibration appears as navigation errors.

Section 5 develops and presents the models for a triad of single degree-of-freedom, floated, rate integrating gyros (operated in a rate mode) or ring laser gyros and the related compensation, which latter is performed in the simulated navigation computer. The dynamics of the SDOF gyros are derived from first principles, then reduced to algorithmic form, where the user is given the choice of three dynamic models varying in effective bandwidth. The treatment of component alignment, is rather complex, and despite the explanation (in Appendix D), must be approached with caution. The laser gyros and compensation models are completely new and ~~the~~ are based mainly on test data. They are vastly simplified relative to their predecessors in the earlier versions of the program. The user may select either type of gyro by appropriate module substitution.

Section 6 develops and presents the models for a triad of strapdown, pendulous, single axis, viscous damped accelerometers with integrated, digital outputs. The linear acceleration resulting from locating the accelerometers some distance from the center of rotation of the vehicle is included, in addition to the errors sources affecting SDOF, floated accelerometers. Three dynamic models may be selected. The compensation scheme, which is included, is more complete than that appearing in most "operational" mechanization.

Section 7 presents the velocity-attitude computations for a strapdown inertial navigator. Since the relevant equations and algorithms were developed in (3) and (5), there was no necessity to develop them from first principles. Updating of the alignment quaternion or direction cosine matrix (d.c.m) by a first, second, or third order algorithm is permitted, with (ortho) normalization at a user-specified frequency. Regardless of the form of update employed, a d.c.m. is formed, since it is required in attitude extraction. A separate, mid-computation cycle d.c.m. is computed for incremental velocity transformation.

Section 8 presents the navigation computations for a strapdown INS, wherein incremental velocity is summed in an inertial frame, then transformed to a LVWA frame for integration. The navigation algorithm corresponds closely to the upgraded algorithm of (5), except for the incremental velocity summation in the inertial rather than the LVWA frame. Attitude is extracted from the appropriate matrix product. Third order vertical channel damping is employed using the simulated barometric altimeter input as a reference.

Section 9 briefly describes the error evaluation routine.

Appendix A, extracted from (5) defines single axis rotations and the general rotation matrix.

Appendix B discusses and describes the simplest set of coordinate frames and transformations which could be employed in the simulation of a strapdown INS with a LVWA computational frame, and transformations actually employed.

Appendix C presents the rotation matrix in terms of the direction cosines of the axis of rotation and the angle of rotation and shows the relationship between a d.c.m and a unit quaternion.

Appendix D discusses the inertial component alignment matrices and indicates how the required 117 entries describing component alignment may be generated for the component models and compensation.

SECTION 2
OVERVIEW OF INSS

2.1 The Evolution of the Strapdown Inertial Navigation System

2.1.1 Historical Prologue

Twenty years ago, inertial navigation and guidance systems were in their infancy.

The inertial measuring unit (IMU) contained three single-degree-of-freedom rate integrating gyros and two or three floated pendulous accelerometers or pendulous integrating gyro-accelerometers (PIGA). Between the inertial sensors and the IMU case were three to five gimbals, each with its own gimbal torque motor (usually geared, two-phase) and one or more resolvers and/or synchros.

The navigation computer was inevitably analog, generally with electromechanical integrating servos, employing motors with tachometer-generator feedback. The platform resolvers were frequently part of the analog computer. Vacuum tubes were being replaced by transistors; the silicon power transistor was brand new. Coordinate transformations (if required) employed resolver chains. The first airborne, transistorized, digital computer - with a digital differential analyser - was in the offing.

The mechanization of choice, for terrestrial (aircraft) applications, was the "analytic" or local-level, north-pointing system, wherein three of the gimbal synchros provided direct indication of the aircraft attitude - roll, pitch and heading (in the five gimbal "geometric" system, synchros or encoders also provided direct outputs of latitude and longitude).

Two other possible mechanizations were of largely academic interest at this time, the "space stable" (or inertially-stabilized) system and the "gimballess" (or strapdown) system. Both of these mechanizations were awaiting the advent of smaller, faster, more powerful, airborne digital computers and the development of pulse-rebalanced gyros and accelerometers.

2.2 The Dawn of the Strapdown System

With the advent of the sixties, active strapdown system development began in earnest. Some of the advantages of the gimballed mechanizations soon came to be more widely appreciated, namely the isolation of the inertial components from the angular motion environment, by the gimbal servos at low frequencies and by the inertia of the stable element at higher frequencies. Generally, further isolation of the inertial components from angular and linear vibration environment was provided by platform isolators, with natural frequencies on the order of 20 to 50 Hz.

Most of the structural modes of the aircraft are below 20 Hz, so a strapdown system, even with isolators, would be subjected to sizable, coupled, angular and linear vibration inputs at low frequencies, as well as the gross vehicle motion due to maneuvers. Some of the resulting errors are due to the kinematics of the motion experienced by the sensors, as in "coning" and "anisoinertia", and will affect even perfect inertial sensors, while others are the results of the interaction of the environment and the dynamics of the particular sensor and its rebalance loop. Only in very special cases are the effects of the above types of forcing functions on strapdown inertial navigation system performance analytically tractable. Thus the available methods of assessing strapdown inertial component or system performance, short of actual flight, are reduced to environmental test and simulation.

Strapdown systems also impose rather severe computational requirements on their navigation computers. These arise from the necessity of transforming the incremental velocity outputs of the accelerometer from the vehicle or body frame to an inertial or earth-referenced computational frame in which the position and velocity of the vehicle are computed.

Since this transformation includes the total vehicle angular motion due to both maneuvers and vibration (up to several hundreds of degrees per second), it has to be computed very frequently and very accurately. Computational errors in strapdown systems could easily equal or exceed the entire system error budget. Here again, the analytical assessment of computational error propagation in strapdown algorithms generally included only those special cases which were analytically tractable.

These analytical results led to the re-emergence of the quaternion (from the mists of antiquity) as the preferred means of generating the transformation from the body (inertial sensor) frame to the computational frame. Very recent indications are that at least one strapdown system manufacturer is returning to the "old" direction cosine or rotation matrix directly.

2.3 The Present Situation

The first strapdown system was flown about ten years ago. Today, there are a number of strapdown systems in varying stages of development and/or test. Flight test results range from low accuracy to moderately high accuracy. With regard to the inertial components, all the accelerometers used are of the pendulous, single-axis variety, generally with analog loops and external analog-to-digital conversion. A full range of gyro types are found ranging from two-degree-of-freedom, free gyros (ESG's) and two-degree-of-freedom, dry-tuned gyros to single-degree-of-freedom floated, rate-integrating gyros and laser gyros, with a variety of gyro signal digitization methods. Only the laser gyro has an inherently digital output.

The digital computers for the strapdown IMU's are changing so rapidly that no general characterization is likely to be correct for long, beyond the fact that the word length is ususally 16 bits and the cycle time is less than one microsecond. They are small, fast, powerful, and cheap, compared with their mini-based predecessors of a few years ago.

2.4 Justification for the Inertial Navigation Strapdown Simulator (INSS)

With the proliferation of existing and projected strapdown INSS's, the need for a flexible, powerful tool for the evaluation of the projected performance of candidate strapdown systems in their operational environment became evident.

It was in an attempt to fulfill this need that the INSS was generated.

2.5 Characteristics of the INSS, or What Does the Simulator Do?

The simulator performs the following functions:

a) Provides a flight profile for a point-mass aircraft executing coordinated maneuvers. Profile includes position, velocity, acceleration and attitude of the aircraft.

b) Superimposes on the flight profile, the angular and linear motion of the aircraft in response to random aerodynamic forces such as gusts and turbulence.

c) Operates on the specific force and angular velocity to produce the ideal body frame values of these quantities and the corresponding angular acceleration.

d) Integrates the gyro and accelerometer equations of motion, of the form specified, after accounting for the displacements of the inertial components from the "center" of the point-mass vehicle, and their orientations with respect to the body frame. Incorporates effects of component parameters and environment on component outputs.

e) Reads true altitude and perturbs the same according to the altimeter model.

f) Transmits sensor data from simulated IMU to simulated navigation computer, and resets sums.

g) Performs accelerometer compensation function (this is essentially the inverse of the accelerometer model, with errors and omissions).

h) Using compensated accelerometer outputs (ΔV's) and gyro outputs ($\Delta\theta$'s) it performs the gyro compensation (again, this is essentially the inverse of the gyro model, with errors and omissions).

i) Updates the body-to-inertial-frame transformation (direction cosine matrix, DCM) and transforms the incremental velocities to the inertial frame (either a DCM or a quaternion update may be employed).

j) Transforms the incremental velocities to the local vertical wander azimuth computational frame and computes local vertical position and velocity (incorporating the barometric altimeter for vertical damping) and attitude.

k) As required, the navigator outputs are differenced with the flight profile values and the resulting errors are printed out and/or plotted.

These functions are shown pictorially in Figure 2-1.

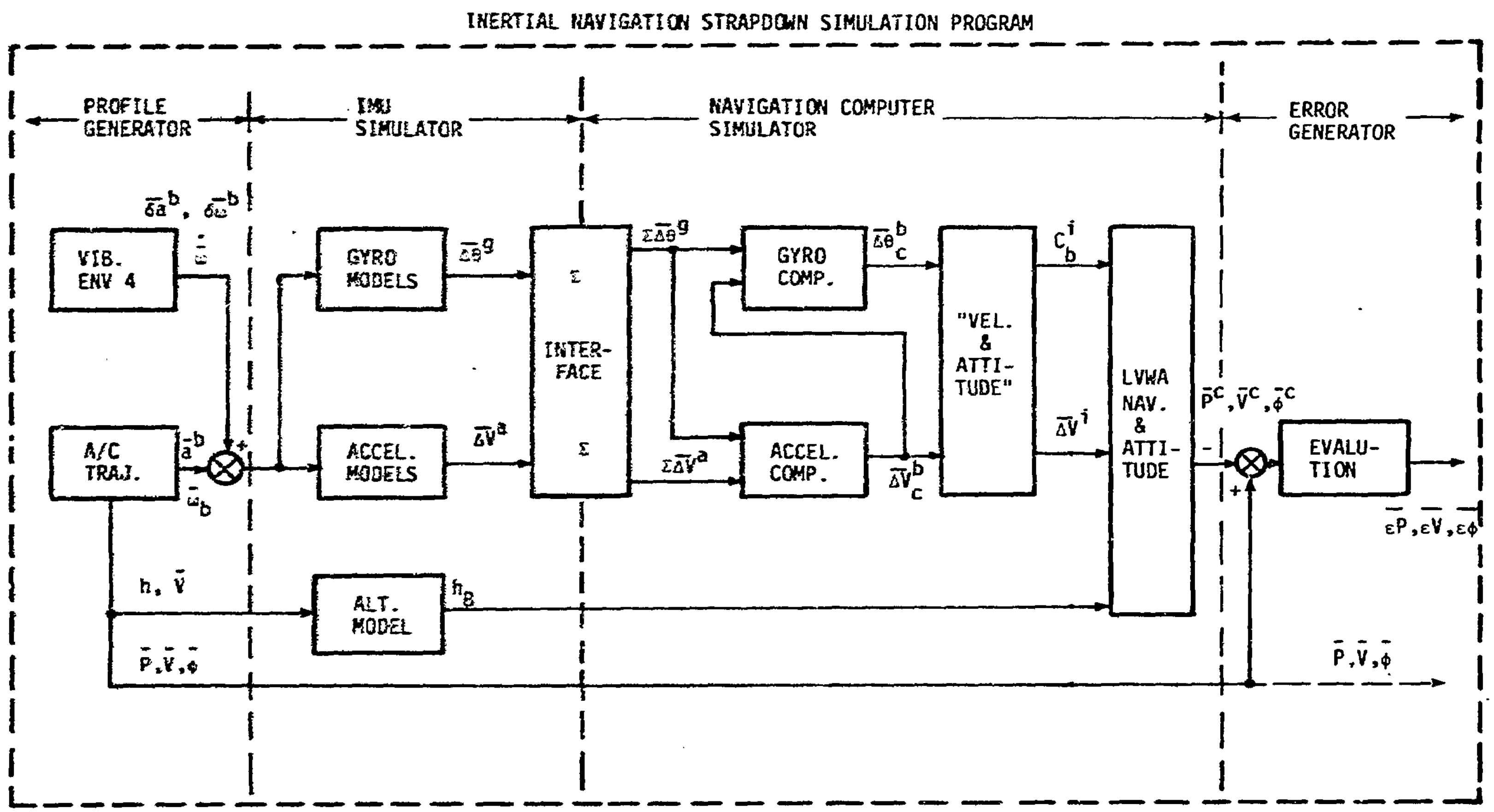


Figure 2-1. INSS FUNCTIONAL BLOCK DIAGRAM

SECTION 3

TRAJECTORY GENERATION

The INSS may employ any trajectory module which is compatible with the conventions used in the remainder of the modules.

The trajectory module which will normally be used to investigate the effects of vehicle dynamics encountered in tactical aircraft is available from the Air Force Avionics Laboratory (RWA-3). This module accepts the outputs of an independent, AFAL-developed, flight profile generator program, PROFGEN(2), which has recently been modified to simulate more closely the dynamics of a point-mass aircraft.

PROFGEN outputs will include the geodetic position (and wander angles if appropriate) earth-relative velocity, and attitude (roll, pitch, and yaw or heading), in addition to specific force (or incremental velocity) in any of several preselected coordinate frames. The earth and gravity models employed in PROFGEN are based on the WGS72 Ellipsoidal Earth Model (1) and hence are compatible with the INSS modules.

The function of the INSS trajectory module, is this case, is largely a matter of accepting the PROFGEN outputs and performing such operations (as initialization, summation, transformation, etc) as are necessary before passing the data on to the remaining INSS modules. The above trajectory module and the flight profile generator program, PROFGEN, are not described further in this report. Instead, the interim trajectory module (which provides the same output data) is described in the remainder of this section.

The trajectory generation module (TRJ) allows the user to simulate elementary flight profiles by specifying a simple set of flight control parameters. The module TRJ was designed as a program driver for software development and debugging; therefore, it has very limited capabilities. It is restricted to constant velocity profiles at fixed aircraft headings. In addition, there is no provision for roll or pitch motion.

The user specifies the initial position, wander angle, and attitude of the aircraft, plus the east, north, up components of velocity relative to the earth.

$$\underline{V} = \left| V_e, V_n, V_u \right|$$

These velocity components are constant throughout the simulated flight. Given the initial conditions, the program computes the current values of:

• Position	L, ℓ, h
• Velocity (constant)	V_e, V_n, V_u
• Heading (constant)	H
• Roll, pitch, yaw (R = const, P = const)	R, P, Y
• Azimuth wander angle	α
• Integrated specific force in body coordinates	$d\underline{v}^b$
• Specific force in body coordinates	$\underline{a}^b$
• Body angular velocity in body coordinates	$\underline{\omega}^b_{ib}$

The specific force and angular velocity are sent to the random environment module (ENV). The altitude and vertical velocity are used in the altimeter module (ALTI). The evaluation module (EVL) compares the values of position, velocity and attitude from the trajectory module with the values computed by the navigation algorithms.

The trajectory module reads two files of input data when it is initialized. During the initialization pass it also generates an output file containing geodetic constants and initialization data which is available to the other program modules. The initialization of the trajectory module is summarized in Section 3.1. The normal module computations are described in Section 3.2.

3.1 Trajectory Module Initialization

The following operations are performed on the initialization pass through the trajectory module.

1. Read the two initialization data files, IFILE and PFILE, and convert the data into internal program units.

IFILE INPUT DATA

Variable	Description	Units
DT	module time step	s
PRNTSW	print control switch	-
OUTSW	not used	-
XFILE	unit number of print file	-
ILAT	geodetic latitude	deg
ILON	geodetic longitude	deg
IALT	altitude	ft
IVEL (1)	ground velocity-east	ft/s
IVEL (2)	ground velocity-north	ft/s

Variable	Description	Units
IVEL (3)	ground velocity- up	ft/s
IPITCH	pitch angle	deg
IYAW	yaw angle	deg
IROLL	roll angle	deg
MODPDT	module print interval	s
IWANDER	initial azimuth wander angle	deg

PFILE INPUT DATA

Variable	Description	Units
WE	earth rotation rate	rad/s
RE	equatorial earth radius	ft
G	nominal gravity	ft/s^2
PRNTDT	general print interval	s

2. Compute the specific force in body coordinates. The specific force in geographic, i.e., (e, n, u), coordinates is

$$\underline{a}^{\ell} = \dot{\underline{V}}_e^{\ell} + (\omega_{en} + 2\omega_{ie}) \times \underline{V}_e^{\ell} - \underline{g}^{\ell}$$

where $\underline{V}_e$ is the velocity relative to the earth and $\underline{g}$ is the gravity vector. The trajectory module is restricted to constant velocity, so $\dot{\underline{V}}_e^{\ell}$ is zero. The program also assumes the velocity $\underline{V}_e$ is zero during the initialization pass and computes

$$\underline{a}^{\ell} = -\underline{g}^{\ell}$$

$$\underline{a}^{b} = C_n^b \underline{a}^{\ell}$$

This is correct only if the earth relative velocity is zero. Fortunately, the accelerometer and gyro modules do not use the initial value of specific force, so the approximation is immaterial.

3. Compute the angular velocity of the body with respect to inertial space in body coordinates. Since the orientation of the body is fixed relative to geographic coordinates, the angular velocity of the body wrt inertial space is equal to the angular velocity of the geographic frame (the ℓ-frame) wrt inertial space.

$$\begin{aligned} \underline{\omega}_{ib} &= \underline{\omega}_{i\ell} \\ &= \underline{\omega}_{ie} + \underline{\omega}_{e\ell} \end{aligned}$$

The angular velocity expressed in inertial coordinates is

$$\underline{\omega}_{ib}^{i} = \begin{bmatrix} 0 \\ w_{ie} \\ 0 \end{bmatrix} + \begin{bmatrix} -(V_n/r_m) \cos \ell \\ V_e/(r_p \cos L) \\ (V_n/r_m) \sin \ell \end{bmatrix}$$

where r_m and r_p are the meridional and prime vertical radii of curvature. The angular velocity is then transformed to body coordinates.

$$\underline{\omega}_{ib}^{b} = C_i^b \underline{\omega}_{ib}^{i}$$

4. Compute the aircraft heading from the specified azimuth wander and yaw angles. The heading remains constant throughout the flight trajectory.

$$H = Y - \alpha$$

where Y is the specified initial yaw angle and α is the initial wander angle.

5. Compute the time rate of change of the roll, pitch and yaw angles. Since the body orientation is fixed relative to geographic coordinates,

$\dot{R} = 0$ (roll rate)

$\dot{P} = 0$ (pitch rate)

$\dot{Y} = \dot{\alpha} = -\frac{V_e}{r_p} \tan L$ (yaw rate) .

6. Generate and write the output data file PFILE. The output data consists of initialization constants from the IFILE and PFILE input data files, plus the initial value of specific force which was computed during the initialization pass. The output PFILE is available to the other program modules. Although the roll, pitch and yaw rates are placed in the PFILE, they are not used by any of modules. All variables in the PFILE are in internal program units.

PFILE OUTPUT DATA

Variable	Description	Units
WE	earth rotation rate	rad/s
RE	equatorial earth radius	ft
$\underline{g}$	nominal gravity	ft/s^2
PRNTDT	general print interval	s
LAT	geodetic latitude	rad
LON	geodetic longitude	rad
WANDER	initial azimuth wander angle	rad
ALT	altitude	ft
ROLL	roll angle	rad
PITCH	pitch angle	rad
YAW	yaw angle	rad
RDOT	roll rate	rad/s
PDOT	pitch rate	rad/s
YDOT	yaw rate	rad/s
VEL(1)	ground velocity-east	ft/s
VEL(2)	ground velocity-north	ft/s
VEL(3)	ground velocity-up	ft/s
AB(1)	specific force $\underline{a}^H$	ft/s^2
AB(2)		ft/s^2
AB(3)		ft/s^2

3.1.1 Position Equations

The program is restricted to constant velocity with respect to the earth in geographic coordinates. The module mechanizes a conventional set of equations to compute latitude, longitude and altitude from the specified velocity (V_e, V_n, V_u). The differential equations are:

$$\omega_{ee} = -V_n/r_m$$

$$\omega_{en} = V_e/r_p$$

$$\omega_{eu} = (V_e/r_p) \tan L$$

$$\dot{L} = -\omega_{ee}$$

$$\dot{\ell} = \omega_{en}/\cos L$$

$$\dot{h} = V_u \qquad (3\text{-}1)$$

where r_m and r_p are the meridional (northerly) and prime vertical (easterly) radii of curvature respectively.

$$r_m = \frac{r_e(1 - \varepsilon^2)}{(1 - \varepsilon^2 \sin^2 L)^{3/2}} + h$$

$$r_p = \frac{r_e}{(1 - \varepsilon^2 \sin^2 L)^{1/2}} + h$$

L	=	geodetic latitude
ℓ	=	geodetic longitude
h	=	altitude
$\underline{V}_e^\ell$	=	$\{V_e, V_n, V_u\}$
$\underline{\omega}_{en}^\ell$	=	$\{\omega_{ee}, \omega_{en}, \omega_{eu}\}$
r_e	=	equatorial earth radius (20925640 ft)

ε = eccentricity

ε^2 = $e(2 - e) = 0.006694317778$

e = $1/298.26$ = ellipticity

The differential equations are solved using rectangular integrations.

$$L_{n+1} = L_n - \omega_{ee}\Delta t$$

$$\ell_{n+1} = \ell_n + (\omega_{en}/\cos L)\Delta t$$

$$h_{n+1} = h_n + V_u \Delta t \qquad (3\text{-}2)$$

3.1.2 Attitude

There is no roll or pitch motion; i.e., $\dot{R}$ and $\dot{P}$ equal zero. The heading is also constant. The yaw angle in the program is defined as

$$Y = \alpha - H$$

so, the yaw rate is equal to the rate of change of the azimuth wander angle.

$$\dot{\alpha} = -\omega_{eu}$$

$$\dot{Y} = -\omega_{eu}$$

Rectangular integration is used to compute the yaw and azimuth wander angles.

$$\alpha_{n+1} = \alpha_n - \omega_{eu}\Delta t$$

$$Y_{n+1} = Y_n - \omega_{eu}\Delta t \qquad (3\text{-}3)$$

The roll, pitch, yaw and azimuth wander angles are used to compute the body attitude matrices C_ℓ^b and C_i^b. The transformations utilize the local level azimuth wander frame (c).

$$C_\ell^b = C_c^b C_\ell^c \qquad (3\text{-}4)$$

$$C_i^b = C_\ell^b C_i^\ell \qquad (3\text{-}5)$$

The C_c^b transformation is a function of the roll, pitch, and yaw angles.

$$C_c^b = X(\Pi)X(R)Y(P)Z(Y)X(\Pi)Z(\tfrac{\Pi}{2})$$

The transformation from (ℓ) to the azimuth wander frame is a simple rotation through the azimuth wander angle.

$$C_\ell^c = Z(\alpha)$$

The details of each of the transformations is explained in Appendix B.

Notice that while the program uses the azimuth wander frame (c), this coordinate system is not needed. It would have been simpler to transform directly to the body frame using the heading angle.

$$C_\ell^b = C(R)C(P)C(H)$$

3.1.3 Angular Velocity of the Body

The average angular velocity of the body over the simulation time step Δt is computed from the Euler rotation of the body over the time interval.

$$\begin{aligned} C_i^b(n)C_b^i(n-1) &= M(\Delta t) \\ &= e^{-F(\underline{\theta}^b)} \\ &\simeq I - F(\underline{\theta}^b) \end{aligned}$$

where

$$F(\underline{\theta}^b) = \begin{pmatrix} 0 & -\theta_{b3} & \theta_{b2} \\ \theta_{b3} & 0 & -\theta_{b1} \\ -\theta_{b2} & \theta_{b1} & 0 \end{pmatrix}$$

The incremental Euler rotation is obtained from the off-diagonal elements of the matrix M; viz.,

$$\underline{\theta}^b = \begin{pmatrix} m_{23} \\ -m_{13} \\ m_{12} \end{pmatrix} \tag{3-6}$$

The average angular velocity in body coordinates is

$$\underline{\omega}_{ib}^b = \underline{\theta}^b/\Delta t \tag{3-7}$$

3.1.4 Specific Force

The specific force in geographic coordinates is computed from the specified velocity. The specific force is then transformed to the body frame. The general expression for the specific force in (e, n, u) coordinates is

$$\underline{a}^\ell = \dot{\underline{v}}_e^\ell + (\underline{\omega}_{e,enu} + 2\underline{\omega}_{iE}) \times \underline{v}_e^\ell - \underline{g}^\ell$$

The program computes the integral of specific force over the simulation time step Δt. The integrated specific force is made available to the evaluation module (EVL) for comparison purposes.

$$d\underline{v}^\ell = \underline{\Delta v}_e^\ell + [(\underline{\omega}_{e\ell} + 2\underline{\omega}_{ie}) \times \underline{v}_e^\ell - \underline{g}^\ell]\Delta t$$

The scalar expressions are

$$\begin{pmatrix} dV_e \\ dV_n \\ dV_u \end{pmatrix} = \begin{pmatrix} \Delta V_e \\ \Delta V_n \\ \Delta V_u \end{pmatrix} + \begin{bmatrix} -(\omega_{eu} + 2\omega_{ie} \sin L)V_n + \omega_{en} + 2\omega_{ie} \cos L)V_u \\ (\omega_{eu} + 2\omega_{ie} \sin L)V_e - \omega_{ee}V_u + g_n \\ -(\omega_{en} + 2\omega_{ie} \cos L)V_e + \omega_{ee}V_n + g_u \end{bmatrix} \Delta t \tag{3-8}$$

where the components of angular velocity with respect to the earth (ω_{ee}, ω_{en}, ω_{eu}) are given in Eq. (3-1). The WGS72 coefficients for the gravity vector are

$$g_n = 1.63 \times 10^{-8}\ h \sin L \cos L$$

$$\begin{aligned} g_u = & -(32.0877057 + 0.16939081 \sin^2 L + 7.52810 \times 10^{-4} \sin^4 L) \\ & \times [1 - (9.6227 \times 10^{-8} - 6.4089 \times 10^{-10} \sin^2 L)\ h + 6.8512 \\ & \times 10^{-15} h^2] \end{aligned}$$

The change in velocity $\underline{\Delta V}_e^{\ell}$ is zero because the specified velocity is constant. The specific force is then computed from $\underline{dV}^{\ell}$ and transformed to body coordinates.

$$\begin{aligned} \underline{a}^{\ell} &= \underline{dV}^{\ell}/\Delta t \\ \underline{a}^{b} &= C_{\ell}^{b} \cdot \underline{a}^{\ell} \end{aligned} \qquad (3\text{-}9)$$

SECTION 4

RANDOM ENVIRONMENT

The environment module (ENV) simulates angular and linear vehicle motion generated by unpredictable sources such as wind buffeting during ground alignment and air turbulence during flight. The current version of the program is restrictive because it allows only one representation of random motion for all segments of the simulated flight profile. This deficiency should be corrected because the environment during ground alignment is drastically different, for example, from the inflight environment of high-performance aircraft.*

Power spectral density plots are usually used to describe random aircraft motions. The shape of the PSDs is usually quite complex. For example, the PSD in Figure 4-1 shows the motions measured in the B-1 radome - forward avionics bay.

The environment module accepts PSD characterizations for each component (roll, pitch, yaw) of linear and angular motion. The module converts the PSD representations into linear and angular random motions as functions of time. In designing the program it was assumed each of the six components of random motion were uncorrelated. While this assumption is certainly not valid, it is convenient because cross-correlation information is normally not available. The program accepts PSDs at the displacement level; i.e.

- Angular displacement PSD - (rad^2/Hz)
- Linear displacement PSD - (ft^2/Hz)

*Alternatively, the capability to run the program in segments e.g. ground alignment, cruise, etc., could be added.

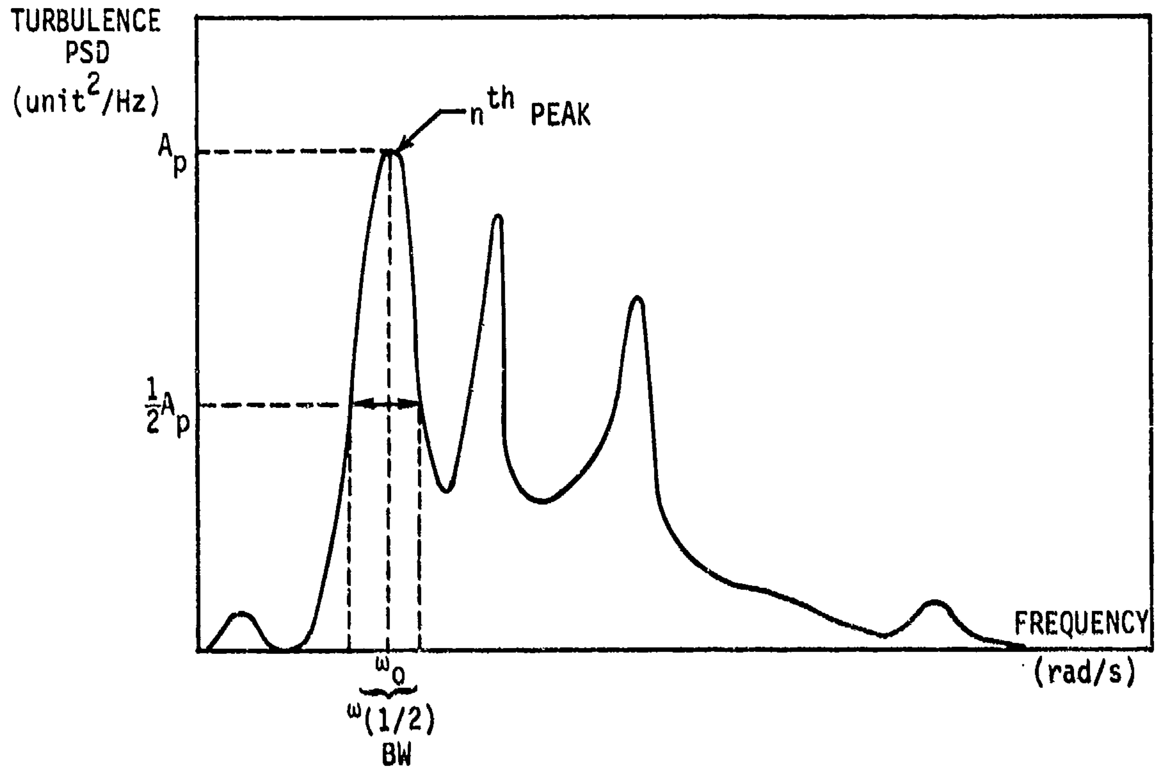


Figure 4-1. Graphic example of PSD parametric specification (turbulence).

The position level PSDs must be constructed if the aircraft motion is specified at the velocity or acceleration level.

To simulate the random motion it is necessary to specify the following items:

a. Descriptions of the Six PSDs

Each PSD is approximated by specifying parameters which describe the PSD in each regions of peak-power density. Three parameters are needed for each region of peak density (see Figure 4-1).

- A_p - peak PSD amplitude (units^2/Hz)
- BW - half power bandwidth (rad/s)
- ω_o - frequency at peak (rad/s)

The number of peaks must also be specified. The program allows a maximum of five peaks for each PSD.

b. Wind-Gust Intensities

The PSD amplitudes specified above represent vehicle motion when the rms gust intensity along each axis is 1 foot/second. The variances of the desired intensity levels must be specified.

• Lateral	$(\text{ft/s})^2$
• Longitudinal	$(\text{ft/s})^2$
• Normal (yaw)	$(\text{ft/s})^2$

c. Simulation Time Step

The simulation time step must be chosen so the simulation frequency is approximately 10 times the highest frequency of interest. The program will generate random motions which approach discrete white noise if the simulation frequency is too low. The highest frequency should be obtained from angular velocity PSDs and linear acceleration PSDs because the position level plots attenuate the high-frequency components.

4.1 Random-Motion Computations

Figure 4-2 shows the operations performed in the random motion module. The linear displacement PSDs are converted into samples of random velocity in body coordinates. The velocity samples are then differenced to form samples of angular accelerations. The angular displacement PSDs are converted into samples of angular velocity in body coordinates. The average angular velocity $\delta\bar{\omega}$ over two simulation cycles is used to compute the transformation from nonvibrating to vibrating body coordinates.

The total linear acceleration is obtained by adding the random acceleration δa^b to the acceleration from the flight-profile generator after it has been transformed to vibrating body coordinates. The total angular velocity is obtained by adding the random angular velocity $\delta\omega^b$ to the angular velocity from the flight-path generator. The derivative of angular velocity is formed by differencing successive samples of total angular rate.

4.1.1 Random-Motion Generator

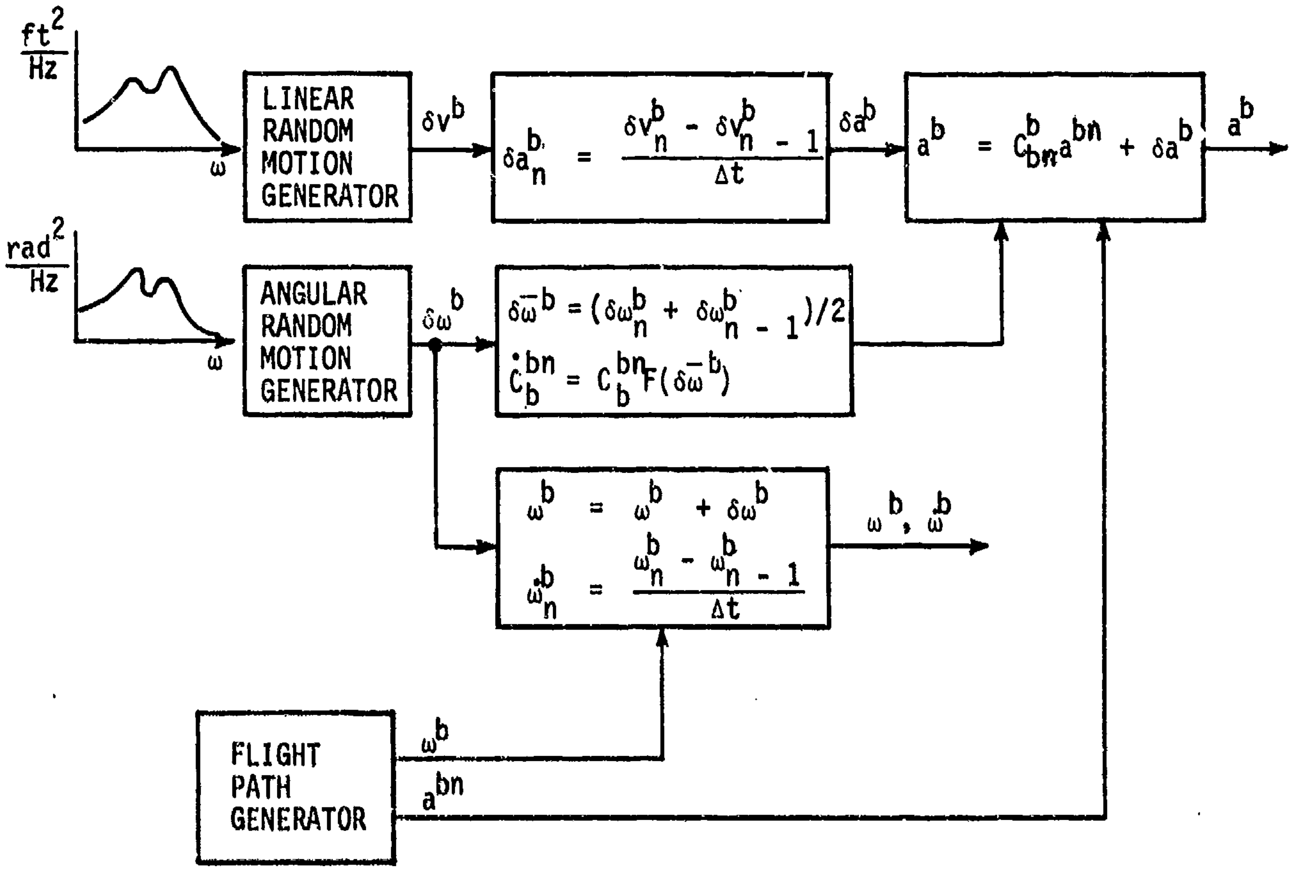
The algorithms for the linear and angular random motion generators are identical — only the units are different. Each displacement PSD is converted into random displacement motion δX samples.

$$\delta v_n = (\delta x_n - \delta x_{n-1})/\Delta t$$

Displacements are simulated rather than velocity or acceleration to guarantee that random displacements will remain bounded in the simulation output.

The equations for generating random motion corresponding to a specified PSD will now be developed. The specified PSD, $S(\omega)$, can be generated by passing continuous white noise $n(t)$ through a shaping filter $H(j\omega)$.

$$S(\omega) = |H(j\omega)|^2 S_n(\omega) \qquad (\text{units}^2/\text{Hz})$$



$\frac{ft^2}{Hz}$
ω
LINEAR RANDOM MOTION GENERATOR
δv^b
$\delta a_n^b = \frac{\delta v_n^b - \delta v_{n-1}^b}{\Delta t}$
δa^b
$a^b = C_{bn}^b a^{bn} + \delta a^b$
a^b
$\frac{rad^2}{Hz}$
ω
ANGULAR RANDOM MOTION GENERATOR
$\delta\omega^b$
$\delta\bar{\omega}^b = (\delta\omega_n^b + \delta\omega_{n-1}^b)/2$
$\dot{C}_b^{bn} = C_b^{bn} F(\delta\bar{\omega}^b)$
$\omega^b = \omega^b + \delta\omega^b$
$\dot{\omega}_n^b = \frac{\omega_n^b - \omega_{n-1}^b}{\Delta t}$
$\omega^b, \dot{\omega}^b$
FLIGHT PATH GENERATOR
ω^b
a^{bn}

Figure 4-2. Random motion module mechanization.

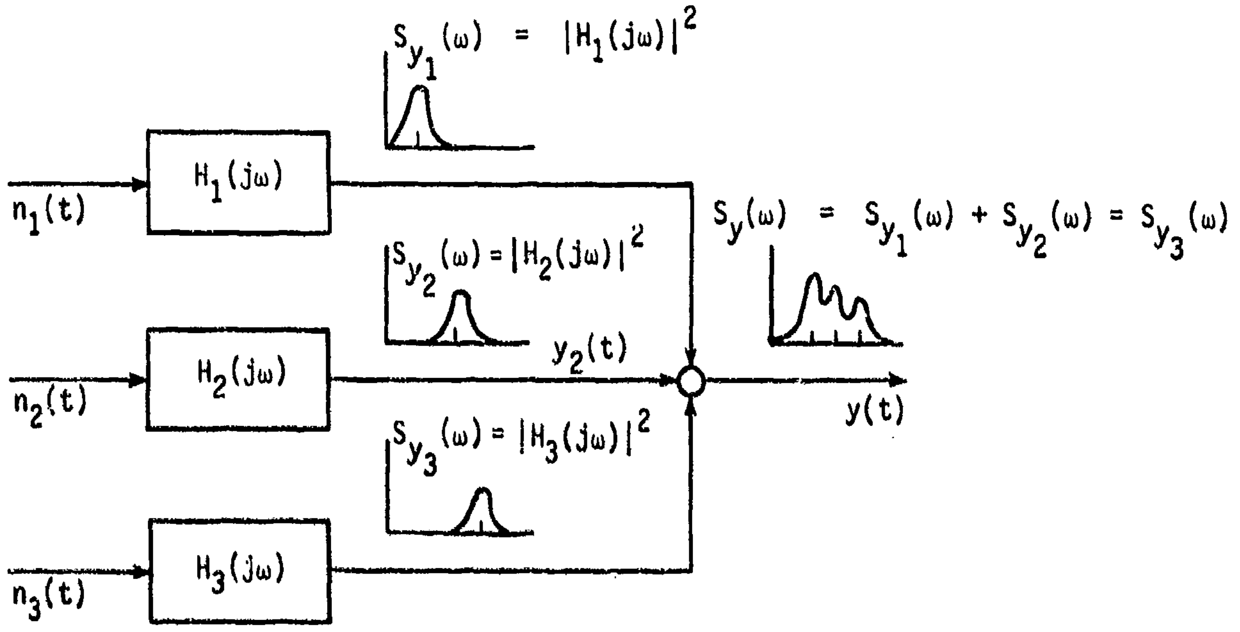
It is convenient to make the white noise dimensionless and of unit amplitude. This allows all of the PSD characteristics to be incorporated into the transfer functions $H(j\omega)$. The power spectral density for unity dimensionless white noise is

$$S_n(\omega) = 1 \qquad (1/\mathrm{Hz})$$

Thus, the PSD of the output of the shaping filter is

$$S(\omega) = |H(j\omega)|^2 \qquad (\mathrm{units}^2/\mathrm{Hz})$$

A PSD, such as the one in Figure 4-1, can be approximated with reasonable accuracy using a set of second-order shaping filters connected in parallel as shown in Figure 4-3. Each peak of the desired PSD is generated by one filter. The input white noise sources are uncorrelated.



$S_{y_1}(\omega) = |H_1(j\omega)|^2$
$n_1(t)$
$H_1(j\omega)$
$S_y(\omega) = S_{y_1}(\omega) + S_{y_2}(\omega) = S_{y_3}(\omega)$
$S_{y_2}(\omega) = |H_2(j\omega)|^2$
$y_2(t)$
$n_2(t)$
$H_2(j\omega)$
$y(t)$
$S_{y_3}(\omega) = |H_3(j\omega)|^2$
$n_3(t)$
$H_3(j\omega)$

Figure 4-3. Generation of complex shaped PSD.

Each of the second-order shaping filters has the transfer function

$$H_i(j\omega) = \frac{\sqrt{k}}{\left(\frac{j\omega}{\omega_n}\right)^2 + 2E\left(\frac{j\omega}{\omega_n}\right) + 1} \tag{4-1}$$

where

ω_n = undamped natural frequency (rad/s)

E = damping ratio

K = gain (units)

The PSD of the filter output is

$$S_i(\omega) = \frac{K}{\left(\frac{\omega}{\omega_n}\right)^4 + 2\left(2E^2 - 1\right)\left(\frac{\omega}{\omega_n}\right)^2 + 1} \tag{4-2}$$

The output PSD is completely determined by K, ω_n, E. These coefficients can be related to the following characteristics of the resonant peak (see Figure 4-1).

A_p - peak amplitude (units^2)

BW - half-power bandwidths (rad/s)

ω_o - frequency at peak (rad/s)

The program user specifies A_p, BW, ω_o for each of the filters. A maximum of five filters can be used to approximate each PSD.

The program computes K, ω_n, E from A_p, BW, ω_o using the following relations:

$$a = \frac{BW}{\omega_o} \quad (0 < a < \sqrt{2})$$

$$\begin{aligned} \omega_n &= \omega_o\left[2 - \left(1 - \frac{a^2}{2}\right)^2\right]^{1/4} \\ E &= \left(\frac{1 - (\omega_o/\omega_n)^2}{2}\right)^{1/2} \\ K &= [1 - (1 - 2E^2)^2]A_p \end{aligned} \qquad (4-3)$$

The second-order shaping filter in Eq. (4-1) will be placed in state variable form to generate the difference equations for computing the sequence of numbers which simulates the random motion. The mechanized equations use the following transform pair to relate the PSD to the auto-correlation functions:

$$S(f) = \int_{-\infty}^{+\infty} R(\tau)e^{-j2\pi f\tau}\, d\tau \quad (\text{units}^2/\text{Hz})$$

$$R(\tau) = \int_{-\infty}^{+\infty} S(f)e^{+j2\pi f\tau}\, df \quad (\text{units}) \qquad (4-4a)$$

where $R(\cdot)$ has the dimensions of $(\text{units})^2$ and S(f) has dimensions of $(\text{units}^2/\text{Hz})$. The transform is frequently written as

$$S(\omega) = \int_{-\infty}^{+\infty} R(\tau)e^{-j\omega\tau}\, d\tau \qquad (\text{units}^2/\text{Hz})$$

$$R(\tau) = \int_{-}^{+} S(\omega)e^{j\omega\tau}\, d\omega \qquad (\text{units}^2) \qquad (4\text{-}4b)$$

Notice that the definition of the power spectrum $S(\omega)$ has not changed so the dimensions are still $(\text{units}^2/\text{Hz})$. Most PSDs (such as the PSD shown in Figure 4-1) are plotted using the single-sided transform

$$S_I(\omega) = 2\int_{0+}^{+\infty} R(\tau)e^{-j\omega\tau}\, d\tau + R(0)\delta(\omega) \qquad \omega \geq 0$$

That is, the PSD is plotted over the positive frequency domain and, except for the dc component, is scaled by a factor of two. The user should scale the single-sided peak amplitude A_p; the program scales A_p to the two-sided transform for the internal calculations. The continuous-state representation for the shaping filter is

$$\dot{\underline{X}}(t) = FX(t) + \zeta\underline{n}(t) \qquad (4\text{-}5)$$

$$Y(t) = HX(t)$$

where $n(t)$ is the dimensionless unit white-noise input to the shaping filter and $Y(t)$ is the filter output. The scalar equations are

$$\begin{pmatrix} \dot{X}_1 \\ \dot{X}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega_n^2 & -2E\omega_n \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \sqrt{K}\omega_n^2 \end{pmatrix} n(t) \qquad (4\text{-}6)$$

$$Y(t) = X_1(t)$$

The mean and covariance of X(t) are

$$E[X(t)] = 0$$

$$P(t) = \phi(t_1 t_0)\, P(t_0)\, \phi(t_1 t_0)^T + \int_{t_0}^{t} \phi(t,\tau)\zeta Q \zeta^T \phi(t,\tau)^T \, d\tau \qquad (4\text{-}7)$$

where

$$P(t) = E[\underline{X}(t)\ \underline{X}(t)^T]$$

$$P(t_0) = E[\underline{X}(t_0)\ \underline{X}(t_0)^T]$$

$$E[n(t)n(\tau)^T] = Q\delta(t - \tau)$$

The magnitude of Q follows from the transform pair.

$$S(\omega) = \int_{-\infty}^{+\infty} Q\delta(\tau)e^{-j\omega\tau}\, d\tau = Q$$

Since n(t) is dimensionless unity white noise

$$Q = 1 \qquad (1/\text{Hz})$$

The discrete time state representation for the shaping filter is

$$\underline{X}(t_{n+1}) = \phi(t_{n+1}, t_n) + B\underline{W}_n \qquad (4\text{-}8)$$

where $\underline{W}_n$ is a vector of dimensionless discrete white-noise samples of unit magnitude. The elements of $\underline{W}_n$ are mutually uncorrelated. Therefore,

$$E[\underline{W}_n \underline{W}_m^T] = I\delta_{mn}$$

The covariance equations for the discrete system is

$$P(t_{n+1}) = \phi(t_{n+1}, t_n)\, P(t_n)\, \phi(t_{n+1}, t_n)^T + Q_n$$

$$Q_n = E[B\underline{W}_n W_n^T B^T] = BB^T \qquad (4\text{-}9)$$

The objective is to simulate the continuous system (Eq. (4-5))

$$\dot{\underline{X}}(t) = FX(t) + \zeta\underline{n}(t)$$

$$Y(t) = HX(t)$$

with the discrete system (Eq. (4-8))

$$\underline{X}_{n+1} = \phi(t_{n+1}, t_n)X_n + B\underline{W}_n$$

$$Y_n = HX_n$$

such that both systems have identical first and second moments. The elements of $\underline{W}_n$ are zero mean random numbers with unity variance. Both systems are zero mean because

$$E[n(t)] = E[\underline{W}_n] = 0$$

To make the covariances in Eq. (4-7) and (4-9) equal, it is necessary to equate

$$BB^T = Q_n = \int_{t_n}^{t_{n+1}} \phi(t,\tau)\zeta\zeta^T \phi(t,\tau)\, d\tau \qquad (4\text{-}10)$$

Thus far, the elements of B are undefined. They will be chosen so Eq. (4-10) is satisfied. The transition matrix over the interval $T = T_{n+1} - t_n$ is

$$\phi(t_{n+1}, t_n) = \phi(T) = \begin{pmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{pmatrix}$$

$$\phi_{11} = 2E\omega_n\phi_{12} + \phi_{22}$$

$$\phi_{12} = \frac{1}{\omega_R} e^{-E\omega_n T} \sin \omega_R T \qquad (4\text{-}11)$$

$$\phi_{21} = -\omega_n^2\phi_{12}$$

$$\phi_{22} = e^{-E\omega_n T} \cos \omega_R T - E\omega_n\phi_{12}$$

where

$$\omega_R = \omega_n\sqrt{1 - E^2}$$

Substituting the expressions for $\phi(t,\tau)$ and ζ in Eq. (4-10) and carrying out the integrations gives

$$Q_n = K\omega_n^4 \begin{matrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{matrix}$$

$$q_{11} = \frac{E^{-2E\omega_n T}\,[E\omega_n \cos(2\omega_R T) - \omega_R \sin(2\omega_R T)]}{4\omega_n^2\omega_R^2}$$

$$+ \frac{1 - e^{-2E\omega_n T}}{4E\omega_n\omega_R^2} - \frac{E}{4\omega_n\omega_R^2}$$

$$q_{22} = \frac{1 - e^{-2E\omega_n T}}{4E\omega_R^2/\omega_n} + \frac{E\omega_n \cos(2b) - \omega_R \sin(2b)}{4\omega_R^2}$$

$$- \frac{e^{-2E\omega_n T}\,[E\omega_n \cos(2\omega_R T + 2b) - \omega_R \sin(2\omega_R T + 2b)]}{4\omega_R^2}$$

$$q_{12} = \frac{e^{-2E\omega_n T} - 1 \sin b}{4E\omega_R^2} + \frac{\omega_R \cos b + E\omega_n \sin b}{4\omega_{R\;n}^2}$$

$$- \frac{e^{-2E\ nT}\,[E\omega_n \sin(2\omega_R T + b) + \omega_R \cos(2\omega_R T + b)]}{4\omega_R^2\omega_n}$$

where

$$b = \sin^{-1}(E)$$

The elements of B are found by equating

$$BB^T = Q_n$$

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{pmatrix} = K\omega_n^4 \begin{pmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{pmatrix}$$

$$b_{11}^2 + b_{12}^2 = K\omega_n^4\, q_{11}$$

$$b_{12}^2 + b_{22}^2 = K\omega_n^4\, q_{22}$$

$$b_{11}b_{21} + b_{12}b_{22} = K\omega_n^4\, q_{12}$$

There are three equations and four b_{ij} coefficients so one coefficient is arbitrary. Let $b_{21} = 0$, then

$$b_{22} = \sqrt{K}\ \omega_n^2 \sqrt{q_{22}}$$

$$b_{12} = \sqrt{K}\ \omega_n^2\, q_{12}/\sqrt{q_{22}} \qquad (4\text{-}13)$$

$$b_{11} = \sqrt{K}\ \omega_n^2 \sqrt{q_{11} - q_{12}^2/q_{22}}$$

This gives all the coefficients needed for the discrete system

$$\underline{X}_{n+1} = \phi(T)\underline{X}_n + \zeta\underline{W}_n$$

$$(4\text{-}8)$$

$$Y_n = H\underline{X}_n$$

The equations used to generate random displacement δX are

$$\boxed{\begin{aligned} X_1(n+1) &= \phi_{11}\, X_1(n) + \phi_{12}\, X_2(n) + b_{11}\, W_1(n) + b_{12}(n) \\ X_2(n+1) &= \phi_{21}\, X_1(n) + \phi_{22}\, X_2(n) + b_{22}\, W_2(n) \\ \delta X(n) &= X_1(n) \end{aligned}} \qquad (4\text{-}14)$$

where $W_1(n)$ and $W_2(n)$ are numbers generated by a zero mean, unity variance Gaussian random number generator. Random velocity is obtained by differencing consecutive samples of δX.

$$\boxed{\delta v(n) = [\delta X(n) - \delta X(n-1)]/T} \qquad (4\text{-}15)$$

The system is stable for all damping ratios greater than zero. Thus, the covariance matrix will always approach steady state. Since the random vehicle motion is assumed to exist prior to the start of the simulation, the initial conditions are chosen so the system starts in the stationary condition. The required relations can be derived directly from the continuous system covariance equation.

$$\dot{P}(t) = FP(t) + P(t)F^T + \zeta Q \zeta^T \qquad (4\text{-}16)$$

In steady state, P(t) is zero.

$$0 = FP_{SS} + P_{SS}F^T + Q\ ^T$$

The F and ζ matrices are given in Eq. (4-6). Q equals unity.

$$\begin{pmatrix} 0 & 1 \\ -\omega_n^2 & -2E\omega_n \end{pmatrix}\begin{pmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{pmatrix} + \begin{pmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{pmatrix}\begin{pmatrix} 0 & -\omega_n^2 \\ 1 & -2E\omega_n \end{pmatrix} + \begin{pmatrix} 0 \\ \sqrt{K\omega_n^2} \end{pmatrix}\begin{pmatrix} 0 & \sqrt{K\omega_n^2} \end{pmatrix}$$

The steady-state covariances are

$$P_{11} = \frac{K\omega_n}{4E}$$

$$P_{12} = 0 \tag{4-17}$$

$$P_{22} = \frac{K\omega_n^3}{4E}$$

The initial state is generated from

$$X(t_0) = A\,\underset{\sim}{W_o} \tag{4-18}$$

where $\underset{\sim}{W_o}$ consists of zero mean unity variance numbers from the Gaussian number generator and the elements of A are chosen so $P(t_0)$ equals P_{SS}.

$$E[\underline{X}(t_0)\ \underline{X}(t_0)^T] = AE[\underset{\sim}{W_o}\underset{\sim}{W_o}^T]A^T$$

$$P_{SS} = AA^T$$

$$\begin{pmatrix} P_{11} & 0 \\ 0 & P_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$$

Solving for a_{ij} terms gives

$$a_{11} = \sqrt{P_{11}} = \sqrt{\frac{K\omega_n}{4E}}$$

$$a_{22} = \sqrt{P_{22}} = \sqrt{\frac{K\omega_n^3}{4E}}$$

$$a_{12} = a_{21} = 0$$

The initial state variables are computed from

$$\boxed{\begin{aligned} X_1(t_0) &= \sqrt{\frac{K\omega_n}{4E}}\; W_1(t_0) \\ X_2(t_0) &= \sqrt{\frac{K\omega_n^3}{4E}}\; W_2(t_0) \end{aligned}} \qquad (4\text{-}19)$$

4.1.2 Random Angular Motion

The output of the random angular motion generator is random angular velocity in body coordinates. The average angular velocity over two consecutive simulation cycles is used to compute the transformation from nonvibratory to vibratory body coordinates.

$$\underline{\delta\omega}^b = \left(\underline{\delta\omega}^b_n + \underline{\delta\omega}^b_{n-1}\right)/2$$

$$C_b^{bn} = C_b^{bn} F \; \underline{\delta\omega}^b$$

A first-order attitude algorithm is used to update the attitude matrix each computation cycle

$$C^b_{bn}(t + T) = M(T)\, C^b_{bn}$$

$$M(T) = \begin{pmatrix} 1 & \delta\omega_3 T & -\delta\omega_2 T \\ -\delta\omega_3 T & 1 & \delta\omega_1 T \\ \delta\omega_2 T & -\delta\omega_1 T & 1 \end{pmatrix}$$

The attitude matrix is orthonormalized each cycle by the following equation which is accurate to first order

$$C_{bn}^{b}(\text{ortho}) = C_{bn}^{b} - \frac{1}{2}C_{bn}^{b}\left[C_{b}^{bn}\,C_{bn}^{b} - I\right]$$

where C_{bn}^{b} is the nonorthonormalization matrix.

The total angular rate in body coordinates is formed by adding the random angular velocity $\delta\omega$ to the angular velocity obtained from the flight-path generator

$$\underline{\omega}^{b} = \underline{\omega}^{b} + \underline{\delta\omega}^{b}$$

where

$\underline{\omega}^{b}$ = total angular velocity in body coordinates

$\underline{\omega}^{bn}$ = angular velocity from flight-path generator

$\underline{\omega}^{b}$ = random angular velocity

The time derivative of total angular rate is obtained by differencing consecutive samples of total angular rate.

$$\dot{\underline{\omega}}_{n}^{b} = \left(\underline{\omega}_{n}^{b} - \underline{\omega}_{n-1}^{b}\right)/T$$

4.1.3 Random Linear Motion

The output of the random linear motion generator is random linear velocity in body coordinates. Random acceleration is computed by differencing consecutive samples of velocity.

$$\underline{\delta a}_{n}^{b} = \left(\underline{\delta v}_{n}^{b} - \underline{\delta v}_{n-1}^{b}\right)/T$$

The total acceleration in the vibrating body frame is obtained by adding the random acceleration δa^B to the acceleration obtained from the flight-path generator after it has been transformed in vibrating body coordinates.

$$\underline{a}^b = C^b_{bn} \, \underline{a}^{bn} + \underline{a}^b$$

where

$\underline{a}^b$	=	total acceleration in vibrating body frame
$\underline{a}^{bn}$	=	acceleration from flight-path generator
$\underline{\delta a}^b$	=	random acceleration
C^b_{bn}	=	transformation from nonvibrating to vibrating body frame

SECTION 5

GYRO MODELS AND COMPENSATION

The program is structured so that either SDF or laser gyro strapdown systems can be simulated. The gyro modules, gyro compensation modules and initialization data files are designed to be functionally identical and can be replaced simply by specifying the correct data sets for the desired class of instruments. The dataset names are listed in Table 5-1.

Table 5-1. Dataset names for SDF and laser gyro modules.

Source Programs and Data Files	DATASET NAMES	
	SDF Gyros	Laser Gyros
Gyro model	SDFGYRO.FORT	RLGGYRO.FORT
Gyro compensation	SDFCOMP.FORT	RLGCOMP.FORT
Gyro model data	SDFGYRO.DATA	RLGGYRO.DATA
Gyro compensation data	SDFCOMP.DATA	RLGCOMP.DATA

The SDF gyro and gyro compensation modules are described in Section 5.1. The RLG laser gyro and gyro compensation modules are described in Section 5.2.

5.1 SDF Gyro Module

The SDF gyro module (gyros) simulates three body mounted single-degree-of-freedom gyros operating in torque-to-balance modes. The single-degree-of-freedom gyro contains a multitude of possible error sources when it is used in strapdown mechanization due to the high angular rates which may be present. Only the most dominant of these error terms—anisoinertia, cross coupling, and output axis accelerations—are included in the model. Float suspension and float misalignment effects are ignored. The gyro model contains the following error sources:

- Anisoinertia
- Spin-output axis cross coupling
- Output axis acceleration
- Gyro input axis misalignment
- Scale-factor error (both positive and negative)
- Scale-factor rate sensitivity (both positive and negative)
- Gyro bias
- Exponentially correlated random drift
- Turn-on transient
- G and G-squared coefficients
- Angular quantization

The program allows the user to specify one of three models for the rebalance loop:

a. "Performance" model which ignores gyro inertia and gyro damping.

b. A first order differential equation model which contains gyro damping.

c. A second order differential equation model which contains both gyro damping and gyro inertial.

The validity of these various approximations will be examined in Section 5.1.1.

Section 5.1.2 derives the equations describing the dynamics of the SDF gyro. The torque-rebalance loop is then constructed using the dynamics of the instrument about its output axis. Section 5.1.3 describes the mechanization of the SDF gyro module.

5.1.1 Dynamics of the SDF Gyro

This section develops the dynamics of the SDF gyro to the extent they are represented in the gyro module. While the dominant characteristics of the gyro are included in the model other known mechanisms have been omitted. The following assumptions are implicit in the derivation:

a. The float is perfectly aligned with the gyro case and can rotate relative to the case only about the output axis. This implies infinite rotational suspension stiffness.

b. The products of inertia of the float assembly are zero.

c. The gyro rotor gimbal is perfectly rigid relative to the float.

d. The gyro rotor is maintained at a constant angular velocity even in the presence of angular vibrations.

The development of the gyro dynamics will closely follow that of Britting [4].

A simplified illustration of the SDF gyro is presented in Figure 5-1, where the input, output, and spin (i, o, s) axes form an orthogonal set fixed to the gyro case. The gyro dynamics are obtained by applying Newton's second law to the float plus rotor assembly.

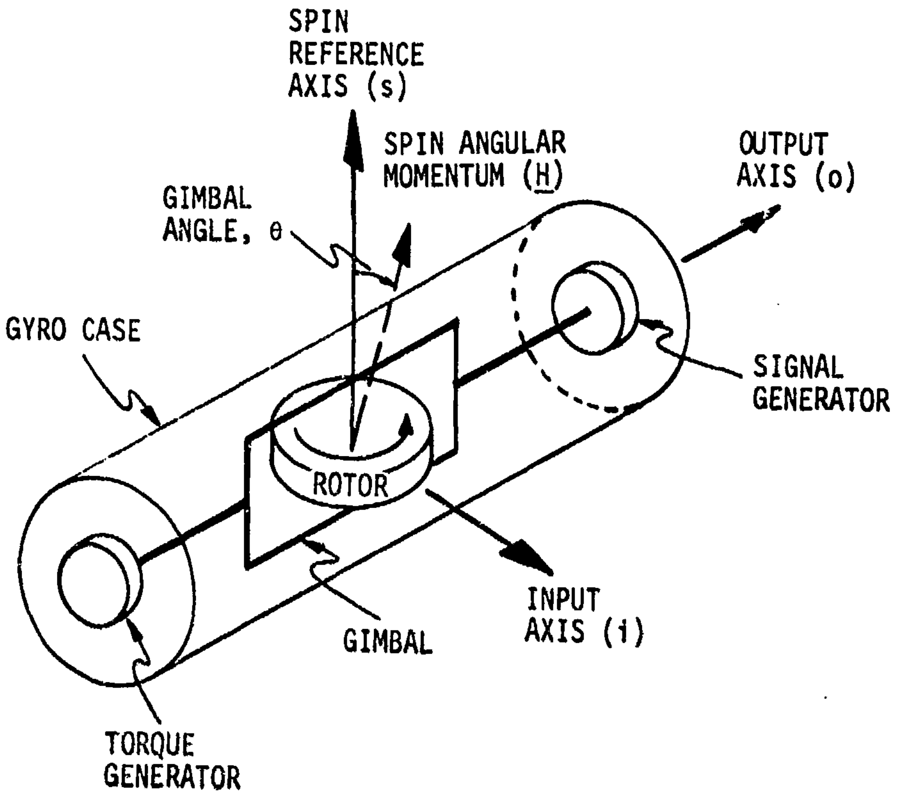


Figure 5-1. Single-degree-of-freedom gyro.

$$\underline{M}_f = \left(\frac{d}{dt}\ \underline{H}_f\right)_i \qquad (5\text{-}1)$$

where

$\underline{M}_f$ = torque applied to the float assembly about the center of mass

$\underline{H}_f$ = angular momentum of the float assembly about the center of mass

$\left(\frac{d}{dt}\ \underline{H}_f\right)_i$ = time derivation of $\underline{H}_f$ wrt inertial space

The time derivative of $\underline{H}_f$ wrt the float is more convenient. The Coriolis relation gives

$$\underline{M}_f = \left(\frac{d}{dt}\underline{H}_f\right)_f + \underline{\omega}_{if} \times \underline{H}_f$$

Expressed in float coordinates (f)

$$\underline{M}_f^f = \dot{\underline{H}}_f^f + F\left(\underline{\omega}_{if}^f\right)\underline{H}_f^f \qquad (5\text{-}2)$$

where $F(\omega)$ is the matrix form of the vector product operator.

$$F(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

The objective is to relate the float dynamics to the angular velocity of the gyro case. Expressing the angular velocity of the float as the angular velocity of the case plus the angular velocity of the float wrt the case gives

$$\underline{\omega}_{if}^f = \underline{\omega}_{ic}^f + \underline{\omega}_{cf}^f$$

$$= C_c^f\,(\underline{\omega}_{ic}^c + \underline{\omega}_{cf}^c) \qquad (5\text{-}3)$$

where

$$\underline{\omega}_{ic}^c = (\omega_i, \omega_o, \omega_s)$$

According to the initial assumptions, the float can rotate relative to the case only about the output axis; i.e.

$$\underline{\omega}_{cf}^{c} = \{0, \dot{\theta}_o, 0\} \qquad (5\text{-}4)$$

and the principal axes of the float assembly are perfectly aligned with the (i, o, j) case axes when the float output angle is zero. Thus, the transformation from case to float is

$$C_c^f = \begin{bmatrix} 1 & 0 & -\theta_o \\ 0 & 1 & 0 \\ \theta_o & 0 & 1 \end{bmatrix} \qquad (5\text{-}6)$$

The angular velocity of the float in float coordinates is

$$\underline{\omega}_{if}^{f} = \begin{bmatrix} \omega_i - \theta_o \omega_S \\ \omega_o + \dot{\theta}_o \\ \omega_i + \theta_o \omega_i \end{bmatrix} \qquad (5\text{-}7)$$

The angular momentum of the float assembly is equal to the angular momentum of the float gimbal plus the angular momentum of the rotor.

$$\underline{H}_f^f = H_g^f + H_R^f$$

$$= I_g \underline{\omega}_{if}^f + H_R^f$$

Since the products of inertia are assumed to be zero,

$$\begin{bmatrix} H_i \\ H_o \\ H_j \end{bmatrix} = \begin{bmatrix} I_i & & \\ & I_o & \\ & & I_S \end{bmatrix} \begin{bmatrix} \omega_i - \theta_o \omega_S \\ \omega_o + \dot{\theta}_o \\ \omega_S + \theta_o \omega_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} \qquad (5\text{-}8)$$

where

H	=	rotor spin angular momentum
I_i, I_o, I_S	=	principal float moments of inertia

Substituting Eq. (5-7) and (5-8) into

$$\underline{M}_f^f = \dot{\underline{H}}_f^f + F(\underline{\omega}_{if}^f)\underline{H}_f^f \qquad (5\text{-}2)$$

gives the following expression for the dynamics about the output axis.

$$M_o = I_o\ddot{\theta}_o - H\omega_i + (I_S - I_j)\omega_i\omega_S + \theta(I_i - I_S)(\omega_i^2 - \omega_S^2)$$
$$+ \theta H\omega_j + I_o\dot{\omega}_o \qquad (5\text{-}9)$$

The torque applied about the output axis, M_o, is assumed to consist of

1. Torque from the torque generator, M_{tg}
2. Viscous torque opposing output axis rotation, $C_o\dot{\theta}_o$
3. Gyro disturbance torques, M_d

That is,

$$M_o = M_{tg} - C_\theta \dot{\theta}_o + M_d \qquad (5\text{-}10)$$

Substituting Eq. (5-10) into Eq. (5-9) are rearranging terms gives

$$\begin{aligned}
I_o \ddot{\theta}_o + C_o \dot{\theta}_o = \; & H\omega_i + M_{tg} && \text{(ideal gyro equations)} \\
& + (I_s - I_i)\omega_i \omega_s && \text{(anisoinertia torque)} \\
& + \theta(I_s - I_i)(\omega_i^2 - \omega_s^2) && \text{(anisoinertia coupling)} \\
& - \theta H \omega_s && \text{(cross-coupling torque)} \\
& - I_o \dot{\omega}_o && \text{(output axis angular acceleration)} \\
& + M_d && \text{(disturbance torque}
\end{aligned}$$

$$(5\text{-}11)$$

Rebalance loops are designed using the ideal gyro equations

$$I_o \ddot{\theta}_o + C_o \dot{\theta}_o = H\omega_i - M_{tg}$$

Since the other terms are ignored, they produce errors in the indicated angular velocity and must be compensated in the navigation computer. While analog torquing may be used in low accuracy applications, pulse torquing is currently used for inertial navigation systems. A typical rebalance loop is shown in Figure 5-2; each output pulse represents an increment of rotation about the input axis. All of the error torques have been lumped into M_d for convenience.

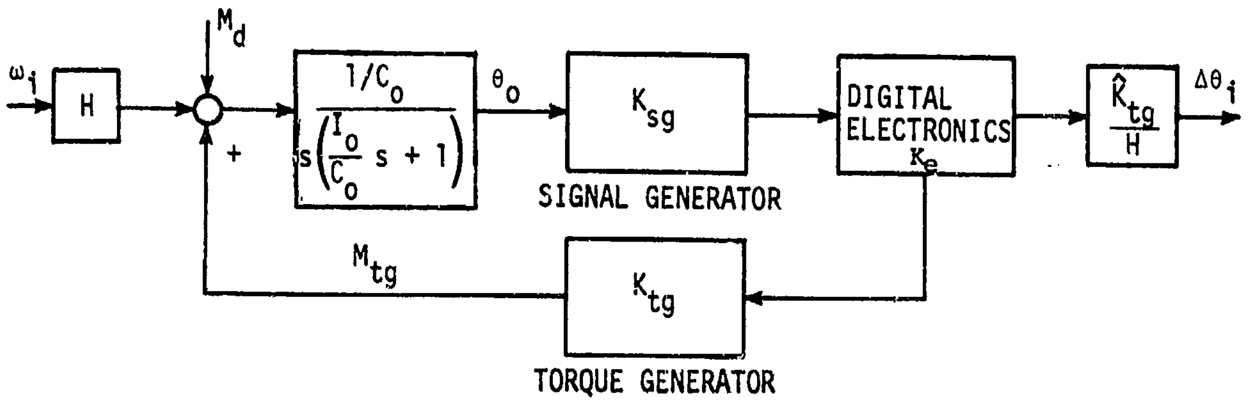


Figure 5-2. Pulse-torquing rebalance loop.

Figure 5-3 shows the rebalance loop mechanized in the gyro module. Notice that the rebalance torque is generated by multiplying the float output angle by the linear gains K where

$$K = K_{sg} K_e K_{tg}$$

Thus, the gyro module simulates an analog rebalance loop followed by a delta-modulator to generate the incremental angle output.

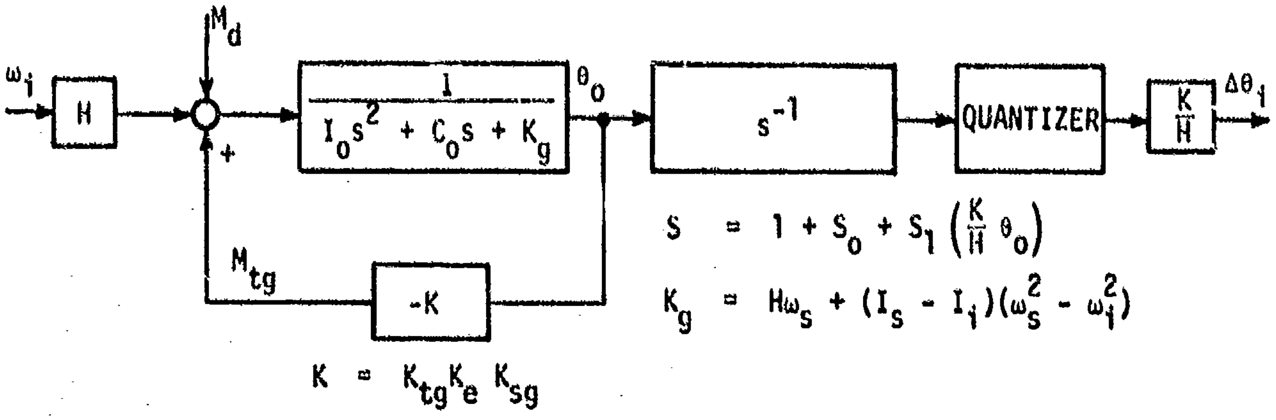

M_d
ω_i
H
+
$\frac{1}{I_o s^2 + C_o s + K_g}$
θ_o
s^{-1}
QUANTIZER
$\frac{K}{H}$
$\Delta\theta_i$
$S = 1 + S_0 + S_1\left(\frac{K}{H}\theta_o\right)$
$K_g = H\omega_s + (I_s - I_i)(\omega_s^2 - \omega_i^2)$
M_{tg}
$-K$
$K = K_{tg}K_e K_{sg}$

Figure 5-3. Simulated analog rebalance loop.

The rebalance loop equations are obtained by substituting

$$M_{tg} = -K\theta_o$$

into Eq. (5-11) and rearranging terms.

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + [H\omega_s + (I_s - I_i)(\omega_s^2 - \omega_i^2) + K]\theta_o = H\omega_i - M_d + (I_s - I_i)\omega_i\omega_s - I_o\dot{\omega}_o$$

Again, it is convenient to lump all of the error terms into the distrubance torque M_d. This gives

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + K_1\theta_o = H\omega_i + M_d^1$$

$$K_1 = H\omega_s + (I_s - I_i)(\omega_s^2 - \omega_i^2) + K \qquad (5\text{-}12)$$

$$M_d^1 = M_d + (I_s - I_i)\omega_i\omega_s - I_c\dot{\omega}_o$$

The gyro module allows the user to selectively solve for θ_o in Eq. (5-12) using one of three models for the rebalance loop:

a. A "performance" model which ignores gyro inertia and gyro damping; i.e.

$$\theta_o = (H\omega_i + M_d^1)/K_1$$

b. a first order differential equations which includes the gyro damping; i.e.

$$C_o\dot{\theta}_o + K_1\theta_o = H\omega_i + M_d^1$$

c. The complete second order representation; i.e.

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + K_1\theta_o = H\omega_i + M_d^1$$

Figure 5-4 shows the effect of approximating the second order differential with first and zero order differential equations. Both approximations artificially increase the gain of the rebalance loop at high frequencies, but the first order approximations is reasonable—it has negligible effect on the bandwidth of the loop. The zero order approximation completely eliminates the loop dynamics and thus produces an infinite bandwidth.

5.1.2 SDF Gyro Module Equations

The SDF gyro module simulates the three rebalance loops sequentially. Figure 5-5 shows the data flow for one of the platform axes—the other two axes are identical. The equations indicated in Figure 5-5 are summarized in this section.

(1) Transformation into Gyro Coordinates

The acceleration, angular rate and derivative of angular rate generated by the random motion module (ENV) is transformed into gyro (i, o, s) coordinates

$$\underline{a}^g = C_B^g\,\underline{a}^B$$

$$\underline{\omega}^g = C_B^g\,\underline{\omega}^B \qquad g = \{g_1, g_2, g_3\}$$

$$\underline{\dot{\omega}}^B = C_B^g\,\underline{\dot{\omega}}^B$$

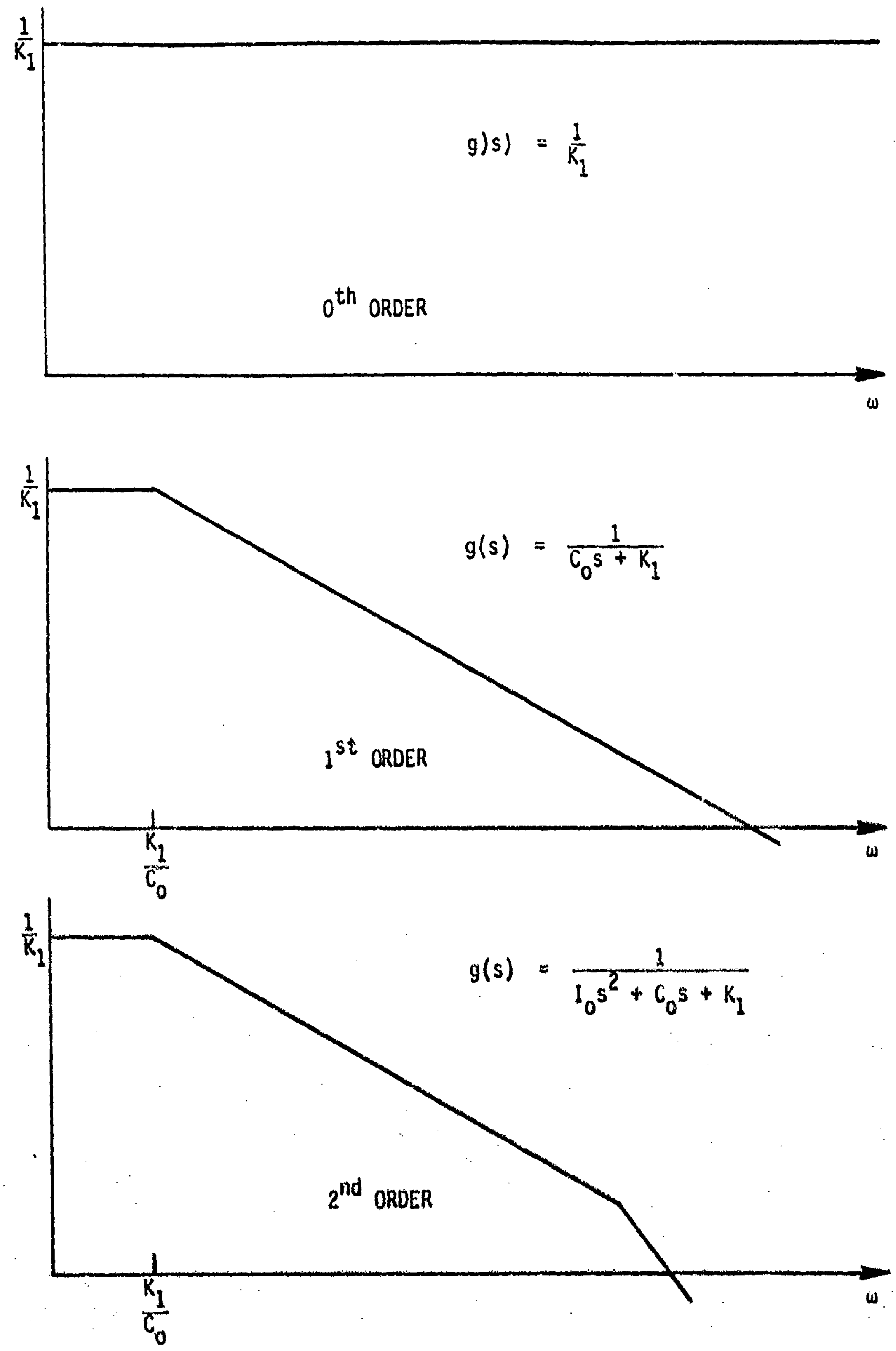


Figure 5-4. Comparison of rebalance loop approximations.

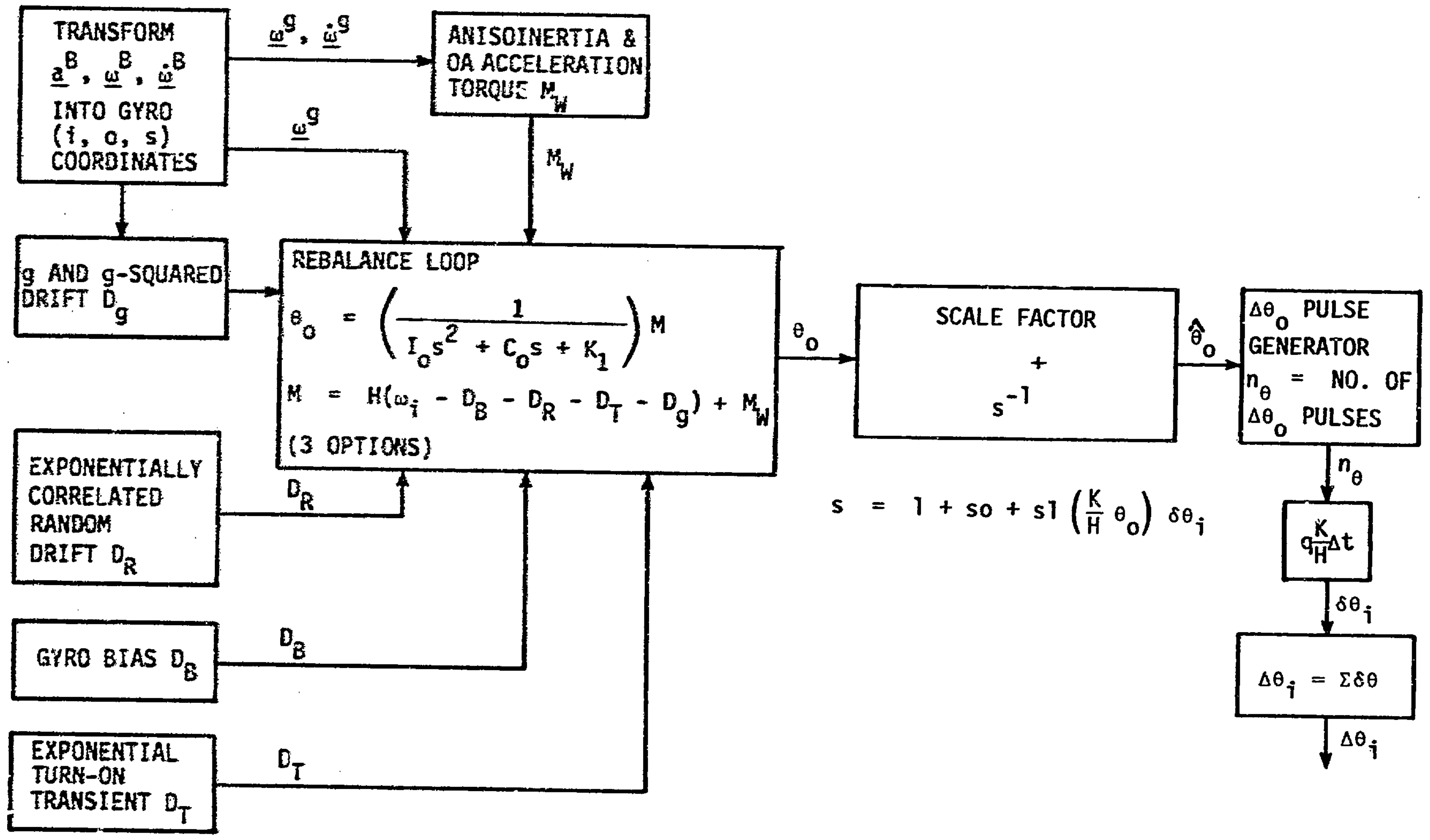

TRANSFORM $\underline{a}^B, \underline{\omega}^B, \dot{\underline{\omega}}^B$ INTO GYRO (i, o, s) COORDINATES
$\underline{\omega}^g, \dot{\underline{\omega}}^g$
ANISOINERTIA & OA ACCELERATION TORQUE M_W
$\underline{\omega}^g$
M_W
g AND g-SQUARED DRIFT D_g
REBALANCE LOOP
$\theta_o = \left(\frac{1}{I_o s^2 + C_o s + K_1}\right) M$
$M = H(\omega_i - D_B - D_R - D_T - D_g) + M_W$
(3 OPTIONS)
θ_o
SCALE FACTOR + s^{-1}
$\hat{\theta}_o$
$\Delta\theta_o$ PULSE GENERATOR n_θ = NO. OF $\Delta\theta_o$ PULSES
n_θ
$q\frac{K}{H}\Delta t$
$\delta\theta_i$
$\Delta\theta_i = \Sigma\delta\theta$
$\Delta\theta_i$
$s = 1 + so + s1\left(\frac{K}{H}\theta_o\right)\delta\theta_i$
EXPONENTIALLY CORRELATED RANDOM DRIFT D_R
D_R
GYRO BIAS D_B
D_B
EXPONENTIAL TURN-ON TRANSIENT D_T
D_T

Figure 5-5. Gyro module computations for each platform axis.

where C_B^g is the transformation to (i, o, s) gyro coordinates. The elements of C_B^g are not programmed constants—they are elements in the gyro initialization data set. The gyro input axis misalignment must be incorporated into the C_B^g input data.

(2) g and g-Squared Drift

The drift generated by the g and g-squared coefficients is obtained by multiplying each coefficient by the appropriate component of $\underline{a}^g$.

$$D_g = K_i a_i + K_o a_o + K_s a_s + K_{ii} a_i^2 + K_{io} a_i a_o + K_{is} a_i a_s + K_{os} a_o a_s + K_{ss} a_s^2$$

where

$\underline{a}^g = \{a_i, a_o, a_s\}$

K_k = linear acceleration sensitivity (mass unbalance) for acceleration along the K^{th} gyro axis

K_{jk} = q-squared sensitivity (compliance) for acceleration along the j and k gyro axes

(3) Gyro Bias

The gyro bias coefficient D_B represents the constant disturbance torque applied to the float. Typically, there is a shift in this coefficient each time the system is turned on.

(4) <u>Exponentially Correlated Random Drift</u>

Exponentially correlated random drift is used to describe the stability of the instrument during periods of continuous operation; i.e., when there are no power cycles. The amplitude and correlation time of the random component are obtained from power spectral density plots of instrument test data. The PSD and correlation functions are related by the transform pair

$$S(\omega) = \int_{-\infty}^{+\infty} R(\tau)\, e^{-j\omega\tau}\, d\tau$$

$$R(\tau) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} S(\omega)\, e^{j\omega\tau}\, d\omega$$

The correlation function for exponentially correlated random drift is uniquely defined by the amplitude and break frequency of the corresponding PSD. Figure 5-6 shows the relations in terms of a single-sided PSD plot which is how most PSD data is displayed.

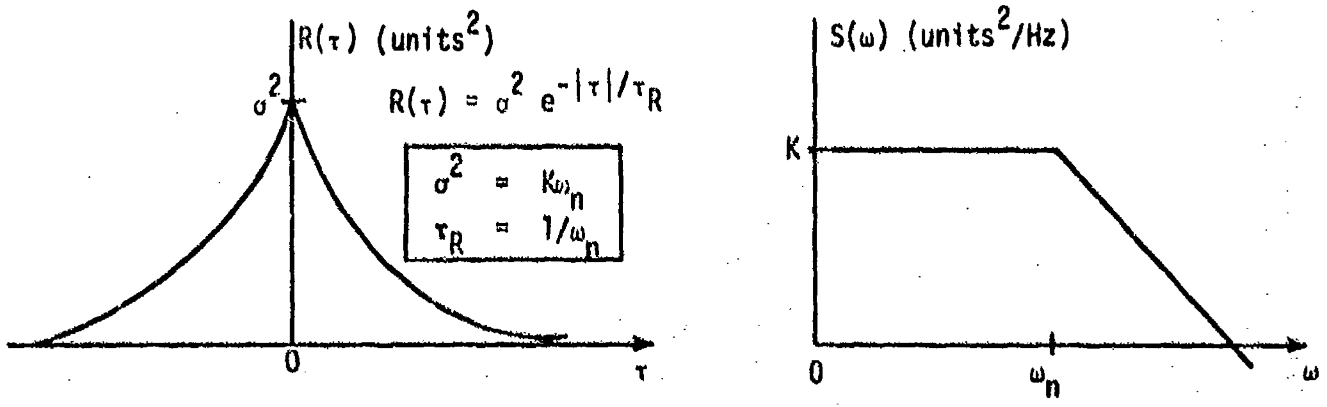


Figure 5-6. Autocorrelation function and PSD for exponentially correlated drift.

The gyro module simulates an exponentially correlated time series using

$$D_R(n+1) = D_R(n)e^{-\Delta t/\tau_R} + \sigma^2 (1 - e^{-2\Delta t/\tau_R}) W_R ; D_R(0) = 0$$

where

D_R	=	simulated random drift
Δt	=	simulation time step
τ_R	=	correlation time
σ^2	=	steady state variance of the random drift
W_R	=	zero mean, unit variance Gaussian sample from the random number generator

Since the random drift is initiated to zero, the mean square drift approaches the steady-state variance in an exponential manner.

$$\sigma_D^2(t) = \sigma^2(1 - e^{-2t/\tau_R})$$

(5) Exponential Turn-On Transient

The spurious error torques generated at turn-on until the thermal system stabilizes is modeled as a simple exponential

$$D_T = D_T(0)\ e^{-t/\tau_T}$$

where

$D_T(0)$	=	initial amplitude of the transient
τ_T	=	exponential decay time constant

(6) Anisoinertia and OA Acceleration Torques

The sum of the anisoinertia and output axis acceleration torques is

$$M_W = (I_s - I_i)\omega_i\omega_s - I_o\dot{\omega}_o$$

where

I_i, I_o, I_s	=	principal inertias of the float about (i, o, s)
ω_i, ω_s	=	angular velocity of the case about the input and spin axes
$\dot{\omega}_o$	=	time derivative of the angular rate of the case about the output axis

(7) Rebalance Loop Dynamics

The float output angle θ_o may be computed using one of the three options shown below.

$$\underbrace{I_o\ddot{\theta}_o + \underbrace{C_o\dot{\theta}_o + \underbrace{K_1\theta_o = M}_{\text{PERFORMANCE MODEL}}}_{\text{FIRST-ORDER MODEL}}}_{\text{SECOND-ORDER MODEL}}$$

$$K_1 = H\omega_s + (I_s - I_i)(\omega_s^2 - \omega_i^2) + K$$

$$M = H(\omega_i - D_B - D_R - D_T - D_g) + M_W$$

where

D_B	=	bias drift
D_R	=	exponentially correlated random drift
D_T	=	turn-on transient
D_g	=	g and g-squared sensitive drift
M_W	=	anisoinertia and OA acceleration torques

This is a direct mechanization of the rebalance loop dynamics shown in Eq. (5-12). The program defines gyro bias, random drift, g and g-squared drift with the opposite sign from the normal conventions. The impact of approximating the rebalance loop with either the "performance" or first order models is discussed in Section 5.1.2. The discrete equations for each options are:

(a) Performance Model

$$\theta_o(n+1) = M(n+1)/K_1$$

(b) First Order Model

$$\theta_o(n+1) = \theta_o(n) + \frac{1}{C_o}[-K_1\theta_o(n) + M(n+1)]\Delta t$$

$$\theta_o(o) = H\omega_i(o)/K$$

(c) Second-Order Model

$$\theta_o(n+1) = \theta_o(n) + \dot{\theta}_o(n)\ t$$

$$\dot{\theta}_o(n+1) = \theta_o(n) + \frac{1}{I_o}\ [-C_o\dot{\theta}_o(n) - K_1\theta_o(n) + M(n+1)]\ T$$

$$\dot{\theta}_o(o) = 0$$

$$\theta_o(o) = H\omega_i(o)/K$$

(8) Torquer Scale Factor

The torquer scale factor consists of a linear term plus a term proportional to input angular rate.

$$S = [1 + SO + S1(\omega_i)]$$

$$= [1 + SO + S1\left[\left(\frac{K}{H}\ \theta_o\right)\right]$$

The scale factor coefficients may have different values for positive and negative angular rates. The scale-factor equations are:

$$S_{(+)} = \left[1 + SO_{(+)} + S1_{(+)}\left(\frac{K}{H}\ \theta_o\right)\right] \quad \theta_o \geq 0$$

$$S_{(-)} = \left[1 + SO_{(-)} + S1_{(-)}\left(\frac{K}{H}\ \theta_o\right)\right] \quad \theta_o < 0$$

where

$SO_{(+)}, SO_{(-)}$	=	positive and negative linear scale factors
$S1_{(+)}, S1_{(-)}$	=	positive and negative rate sensitive scale factors

(9) Angular Quantization

The pulse generator computes the number of output pulses produced by the output angle θ_o plus the stored angle residual.

$$\theta = \theta_o(n+1) + \theta_r(n) \quad \theta_r(0) = 0$$

$$\text{If } (\theta \geq 0) \; S = 1 + SO_{(+)} + S1_{(+)} \left(\frac{K}{H}\,\theta\right)$$

$$\text{If } (\theta < 0) \; S = 1 + SO_{(-)} + S1_{(-)} \left(\frac{K}{H}\,\theta\right)$$

$$n_\theta = \text{Integer } \frac{\theta}{qS}$$

$$\hat{\theta}_o = qSn_\theta$$

$$\theta_r(n+1) = \theta - \check{\theta}_o$$

$\check{\theta}_o$	=	quantized indication of float output angle
θ_o	=	float output angle
θ_r	=	quantized angle residual
S	=	scale factor
q	=	$\Delta\theta_o$ quantization level
n_θ	=	number of $\Delta\theta_o$ pulses

The output pulses are then converted into an indication of the angular rotations about the input axis of the gyro. The output angle is a direct indication of the angular velocity.

$$\begin{aligned} \omega_i &= \frac{K}{H} S^{-1} \ddot{\theta}_o \\ &= \frac{K}{H} qn_\theta \end{aligned}$$

Thus, the measured rotations about the input axis is

$$\begin{aligned} \delta\theta_i &= \hat{\omega}_i \Delta t \\ &= qn_\theta \frac{K}{H} \Delta t \end{aligned}$$

$$\Delta\theta_i = \Sigma\delta\theta_i$$

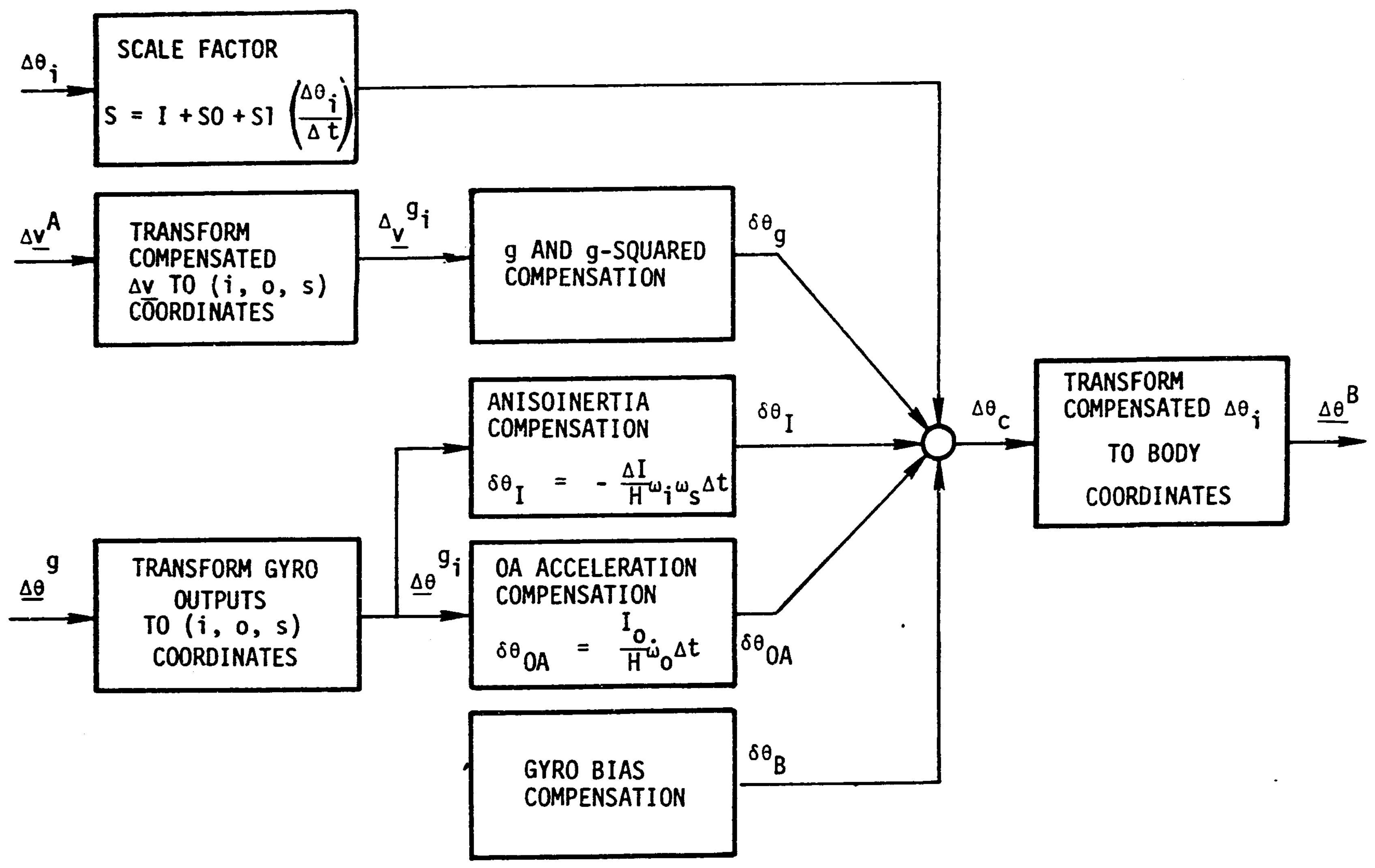
The navigation equations reset $\Delta\theta_i$ to zero each time the platform attitude matrix is computed.

5.1.3 SDF Gyro Compensation

The SDF gyro compensation module (GCOMP) computes the compensation for the following error sources:

- Gyro input axis misalignment.
- Scale factor error (positive and negative).
- Scale factor rate sensitivity (positive and negative).
- Bias drift.
- g and g-squared sensitive drift.
- anisoinertia torque.
- Output axis acceleration torque.

A flow diagram of the compensation module is shown in Figure 5-7. The compensation equations for each platform axis are summarized below.



$\Delta\theta_i$
SCALE FACTOR
$S = I + S0 + S1 \left(\frac{\Delta\theta_i}{\Delta t}\right)$
$\underline{\Delta v}^A$
TRANSFORM COMPENSATED $\underline{\Delta v}$ TO (i, o, s) COORDINATES
$\Delta\underline{v}^{g_i}$
g AND g-SQUARED COMPENSATION
$\delta\theta_g$
$\underline{\Delta\theta}^g$
TRANSFORM GYRO OUTPUTS TO (i, o, s) COORDINATES
$\underline{\Delta\theta}^{g_i}$
ANISOINERTIA COMPENSATION
$\delta\theta_I = -\frac{\Delta I}{H}\omega_i\omega_s\Delta t$
$\delta\theta_I$
OA ACCELERATION COMPENSATION
$\delta\theta_{OA} = \frac{I_o}{H}\dot{\omega}_o\Delta t$
$\delta\theta_{OA}$
GYRO BIAS COMPENSATION
$\delta\theta_B$
$\Delta\theta_c$
TRANSFORM COMPENSATED $\Delta\theta_i$ TO BODY COORDINATES
$\underline{\Delta\theta}^B$

Figure 5-7. Gyro module computations for each platform axis (Compensation Module)

(1) <u>Scale-Factor Compensation</u>

The incremental platform rotations about each gyro input axis are compensated for linear and rate sensitive scale factors. Separate scale factors for positive and negative angular rates are provided.

$$S = 1 + SO_{(+)} + S1_{(+)} \frac{\Delta\theta_i}{\Delta t} \qquad \Delta\theta_i \geq 0$$

$$S = 1 + SO_{(-)} + S1_{(-)} \frac{\Delta\theta_i}{\Delta t} \qquad \Delta\theta_i < 0$$

where

$SO_{(+)}, SO_{(-)}$	=	positive and negative linear scale factors
$S1_{(+)}, S1_{(-)}$	=	positive and negative rate sensitive scale factors

(2) <u>g and g-squared Sensitive Drift Compensation</u>

The compensated incremental velocities from the accelerometers are transformed into gyro (i, o, s) coordinates.

$$\Delta\underline{v}^g = C_b^g \, \Delta\underline{v}^A$$

where

$\Delta\underline{v}^g$	=	$\{\Delta v_i, \Delta v_o, \Delta v_s\}$ of K^{th} gyro
$\Delta\underline{v}^A$	=	compensated accelerometer outputs
C_b^g	=	transformation from body to the K^{th} gyro

The drift produced by the acceleration sensitive coefficients is

$$\varepsilon\omega_i = -(K_i a_i + K_o a_o + K_s a_s)$$

$$-(K_{ii} a_i^2 + K_{io} a_i a_o + K_{is} a_i a_s + K_{os} a_o a_s + K_{ss} a_s^2)$$

The program computes corrections to the incremental rotation about the gyro input axis.

$$\delta\theta_g = K_i \Delta v_i + K_o \Delta v_o + K_s \Delta v_s$$

$$+(K_{ii} \Delta v_i^2 + K_{io} \Delta v_i \Delta v_o + K_{is} \Delta v_i \Delta v_s + K_{os} \Delta v_o \Delta v_s + K_{ss} \Delta v_s^2)/\Delta t$$

where K_K and K_{jk} are the compensation coefficients.

K_K	=	linear acceleration sensitivity for acceleration along the K^{th} gyro axis
K_{jk}	=	acceleration sensitivity for accelerations along the j and k gyro axes

(3) <u>Gyro Bias Compensation</u>

Gyro bias produces a constant error in indicated angular rate along the gyro input axis.

$$\varepsilon\omega_i = -D_B$$

The bias compensation is

$$\delta\theta_B = D_B \Delta t$$

(4) Anisoinertia Torque Compensation

Anisoinertia torque produces the angular rate error

$$\delta\omega_i = \frac{1}{H}(I_s - I_i)\omega_i\omega_s$$

The anisoinertia torque compensation is

$$\delta\theta_A = -\frac{1}{H}(I_s - I_i)\Delta\theta_i\Delta\theta_s/\Delta t$$

where

H	=	rotor spin angular momentum
I_i, I_s	=	principal moments of inertia

(5) OA Acceleration Torque Compensation

Output axis acceleration torque produces the angular rate error

$$\delta\omega_i = -\frac{I}{H}\dot{\omega}_o$$

The OA acceleration compensation is

$$\delta\theta_{OA} = \frac{I_o}{H}[\Delta\theta_o(n+1) - \Delta\theta_o(n)]/\Delta t$$

where

$\Delta\theta_o$	=	incremental rotations about the output axis
I_o	=	float inertia about the output axis
H	=	rotor spin angular momentum

(6) Total Gyro Compensation

The compensated rotation about the gyro input axis is

$$\Delta\theta_i(\text{comp}) = S(\Delta\theta_i) + (\delta\theta_B + \delta\theta_g + \delta\theta_I + \delta\theta_{OA})$$

where

S	=	scale factor compensation
$\delta\theta_B$	=	bias compensation
$\delta\theta_g$	=	g and g-squared compensation
$\delta\theta_I$	=	anisoinertia torque compensation
$\delta\theta_{OA}$	=	OA acceleration compensation

The three compensated gyro outputs are then transformed from the gyro input axes to body coordinates.

$$\underline{\Delta\theta}^b = C_g^b \, \underline{\Delta\theta}^g$$

where

$\underline{\Delta\theta}^g$	=	compensated gyro outputs along the three gyro input axes
C_g^b	=	(nonorthogonal) transformation from the gyro input axis to body coordinates

The transformation matrix C_g^b is labeled QMIS in the program. The elements of this matrix must be consistent with the rotation matrices which specify the (i, o, s) axes of each gyro relative to body coordinates. These rotation matrices are labeled QGBX, QGBY, and QGBZ. All of the matrices are obtained from the initialization dataset for the compensation module.

5.2 LASER GYRO MODULE

The laser gyro module (GYROS) simulates three body mounted gyros plus their incremental angle detectors. The laser gyro model contains the following error sources:

- Gyro input axis misalignment.
- Scale factor error.
- Scale factor turn-on transient.
- Gyro bias drift.
- Turn-on transient drift.
- Angular random walk.
- White angular noise.
- Angular quantization.

The model of the laser gyro is discussed in Section 5.2.1. Section 5.2.2 describes the mechanization of the laser gyro module (GYROS) and Section 5.2.3 describes laser compensation module (GCOMP).

5.2.1 Principles of Operation and Model of the Laser Gyro

This section develops the principles of operation and other salient characteristics of the ring laser gyro, a recent development in optical technology which shows promise of replacing many conventional inertial gyros. These principles lead, in turn, to the model implemented in this module for simulating the performance of these strapdown laser gyros.

The laser gyro is basically a laser that has three or more reflectors arranged to enclose an area. These three mirrors, in conjunction with the light-amplifying material in the laser path, comprise two optical oscillators—one that has energy traveling clockwise, and one that has energy traveling counterclockwise around the same path.

The optical length of the path they travel determines the frequencies at which these oscillations operate. When the enclosed ring is rotated in inertial space, the clockwise and counterclockwise paths have effectively different lengths. The path difference in these two directions causes the two oscillators to operate at different frequencies, and this difference is proportional to the rate at which the ring is rotating. The readout of the laser gyro is then accomplished by monitoring the frequency difference between the two lasers.

The foundation of laser gyro technology is the description of the phenomenon by which the path around the ring can be different for observers (photons) traveling with the direction of rotation than for observers traveling against the direction of rotation. In accordance with the principles of general relativity, two observers, traveling around a closed path that is rotating in inertial space, will find that their clocks are not in synchronization when they return to the starting point (traveling once around the path, but in opposite directions). The observer traveling in the direction of rotation will experience a small increase, and the observer traveling in the opposite direction will experience a small decrease in clock time. The difference in the readings of these clocks depends upon the inertial rotation rate, the area enclosed by the path, and the speed of light. Since the laser gyro uses photons, traveling at the speed of light, for observers, the time difference appears as an apparent length change in the two paths equal to the observers' velocity times their time difference. Accordingly, even though both observers traverse an identical physical path, they see an apparent length change which is proportional to the enclosed area of the path and its rotation rate in inertial space.

The fundamental boundary condition governing the operation of the laser gyro is that the laser wavelength λ must be equal to an integer function of the optical length L around the cavity. That is

$$\lambda = L/N \qquad (1)$$

where N is a large integer typically in the range of 10^5 to 10^6. A length change of ΔL will cause a wavelength change of

$$\Delta\lambda = \Delta L/N \qquad (2)$$

The corresponding frequency change, by logarithmic differentiation, is given as

$$\frac{\Delta f}{f} = \frac{\Delta L}{L} \qquad (3)$$

Accordingly, given small length differences ΔL and reasonable cavity lengths L, the operating frequency should be as high as possible.

The relationship between inertial input rate Ω and apparent length change ΔL can easily be shown to be

$$\Delta L = \frac{4A\Omega}{C} \qquad (4)$$

where A and C represent the enclosed area and the speed of light, respectively.

Thus, the equation relating Ω to Δf, in terms of gyro size and wavelength, is determined by substituting Eq. (4) into Eq. (3), yielding

$$\Delta f = \frac{4A\Omega}{\lambda L} \qquad (5)$$

This represents the <u>ideal</u> laser gyro equation, but fails to account for the three principal error sources effecting the operation of all laser gyros. In descending order of importance, they are: lock-in, null shift, and mode pulling.

Lock-in arises due to the mutual coupling between the oppositely directed traveling waves and has been the focus for most of the effort to eliminate the laser gyro's error sources. When the rotation rate is reduced below some critical value (called the lock-in threshold), the frequency difference between the oppositely traveling waves synchronizes to a common value. Thus, for rotation rates below the lock-in threshold, the laser gyro is unresponsive to rotations.

The principal method employed to minimize the effect of lock-in is to bias the laser gyro with an artificial input rotation such that zero inertial rotation rate corresponds to a point exterior to the locked region. As a result, the system operates at all times with a differential path and a frequency difference with respect to the oppositely traveling beams. Such a bias can be introduced by any technique which introduces an anisotropy in the index of refraction, again with respect to the oppositely directed beams, and is commonly effected by either mechanical or optical means. In turn, each of these general techniques can be employed in two ways: by holding the bias fixed, measuring the input rate or angle, and observing differences in the laser gyro output rate due to the input rate, or by introducing a negative-to-positive alternating bias.

Due to a reduction in the requirements on the absolute stability of the magnitude of the bias, alternating bias techniques are generally employed. The idea behind this method is to minimize the (switching) time that the laser spends in the locked region. Since the laser gyro is basically an integrating rate instrument, only the net rotation appears in the output. The basic premise in the alternating bias technique is that the bias is perfectly symmetrical. Thus, the laser is allowed to remain unlocked over most of the "dither" cycle, and hence the gyro is responsive to any nonzero input rate.

The error generated when the laser gyro is biased by a sinusoidal dither, at a rate several orders of magnitude greater than the lock-in rate, is characterized by a one-dimensional, random-walk process. Although the mean or expected error is zero, the root mean square error is not and is given by

$$\text{rms error } (t) = \sqrt{\frac{K\Omega_L^2}{4\pi\Omega_D} t} \quad \text{arcsec} \qquad (6)$$

where

K	=	gyro scale factor (arcsec/count)
t	=	time (s)
Ω_L	=	lock-in threshold (deg/h = arcsec/s)
Ω_D	=	peak dither rate (deg/h = arcsec/s)

Numerical values are a function of the gyro size and operative parameters. As an example, the Honeywell GG/1300 laser gyro (5.7 inch, visible) has $K = 1.5$ arcsec/count, Ω_L typically 0.5 deg/s (1800 deg/h), and $\Omega_D = 3.77 \times 10^5$ deg/h (amplitude of 400 arcseconds at a frequency of 150 Hz). Accordingly, for this particular instrument

$$\text{rms error } (t) = 1.01\, t^{1/2} \text{ arcsec}$$

In addition to any errors inherent in the laser gyro, there is additional uncertainty associated with the sampling process.

With regard to other error sources, a null shift (bias) is introduced into the output of the laser gyro by any effect—exclusive of inertial rotation rates—which makes the clockwise optical path length different from the counterclockwise path length. These effects arise from the anisotropy (nonreciprocation) of the optical parameters within the cavity with respect to radiation traveling in two directions and cause a shift of the beat frequency (output) versus rotation rate ideal curve. Unless the gyro is properly designed, null-shift errors can be orders of magnitude greater than input rates.

On the other hand, mode pulling, more correctly known as anomalous dispersion correction, is a phenomenon which causes variations in the gyro scale factor. The difficulty in maintaining scale factor accuracy does not lie in the geometric dimensions, but rather in the index of refraction. The anomalous dispersion due to the atomic transitions in the lasing medium creates a gain-dependent correction to the scale factor which is a strong function of the operating characteristics of the laser.

5.2.2 Laser Gyro Model Equations

The laser gyro module (GYROS) simulates three strapped-down laser gyros. The gyro module computations are shown in Figure 5-8. The equations are summarized in this section. The equations for each platform axis are identical.

(1) Scale-Factor Turn-On Transient

The gyro scale factor consists of a bias plus an initial turn-on transient. The turn-on transient is modeled as a simple exponential function.

$$S_T = S_T(0)e^{-t/\tau_S}$$

where

$S_T(0)$ = initial scale factor transient amplitude

τ_S = exponential transient time constant

(2) Scale-Factor and Gyro IA Misalignment

The three gyro input axes are aligned with the body axes except for the gyro misalignment angles which are relatively small. The transformation from body to gyro IA axes is

$$C_B^g = I + \delta C_B^g$$

The program combines the gyro input axis misalignments with the total scale factor errors.

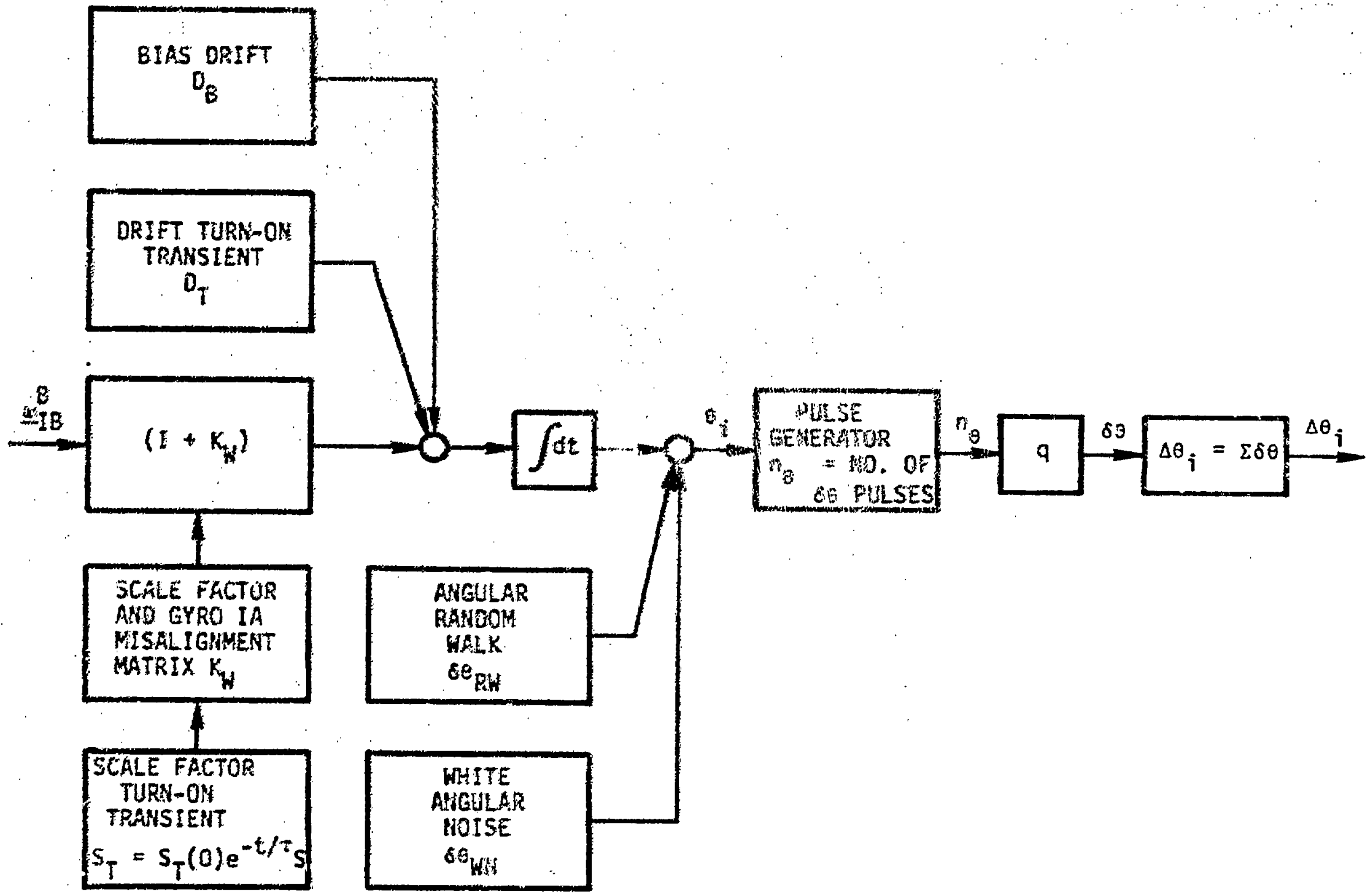

BIAS DRIFT D_B
DRIFT TURN-ON TRANSIENT D_T
$\underline{\omega}^B_{IB}$
$(I + K_W)$
SCALE FACTOR AND GYRO IA MISALIGNMENT MATRIX K_W
SCALE FACTOR TURN-ON TRANSIENT $S_T = S_T(0)e^{-t/\tau_S}$
$\int dt$
ANGULAR RANDOM WALK $\delta\theta_{RW}$
WHITE ANGULAR NOISE $\delta\theta_{WN}$
θ_i
PULSE GENERATOR n_θ = NO. OF $\delta\theta$ PULSES
n_θ
q
$\delta\theta$
$\Delta\theta_i = \Sigma\delta\theta$
$\Delta\theta_i$

Figure 5-8. Laser gyro module computations.

$$K_W = S + \delta C_b^g$$

$$K_W = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{33} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} = \begin{pmatrix} S_1 & \nu_{13} & -\nu_{12} \\ -\nu_{23} & S_2 & \nu_{21} \\ \nu_{32} & -\nu_{31} & S_3 \end{pmatrix}$$

The elements on the main diagonal are total scale factor error.

$S = S_O + S_T$

S_O = constant scale factor error

S_T = exponential turn-on transient

The off-diagonal elements are the input axis misalignments.

ν_{ij} = misalignment of the i^{th} gyro input axis about the j^{th} platform axis

The laser gyro module obtains the values of the K_W elements from the module initialization file. The K_{ij} elements are specified directly rather than the values for scale factor and IA misalignments. The elements on the main diagonal are the values of constant scale factor error.

$$K_{ii} = S_{o_i} \quad \text{(ppm)}$$

The off-diagonal elements correspond to the input axis misalignments except for sign.

$$K_{ij} = \pm\nu_{ik} \quad \text{(microrad)}$$

The program then adds the scale-factor transients to the main diagonal elements to form the complete rate sensitive matrix.

(3) Gyro Bias

The gyro bias D_B produces a constant error in the indicated angular velocity along the gyro's input axis. Typically there is a shift in this coefficient each time the system is turned on.

(4) Exponential Drift Turn-On Transient

The turn-on drift transient is modeled as a simple exponential.

$$D_T = D_T(0)e^{-t/\tau_D}$$

where

$D_T(0)$ = initial amplitude of the drift transient

τ_D = exponential transient time constant

(5) White Angular Noise

The measured angular rotation about the gyro input axis is corrupted by angular random walk and angular noise which is essentially uncorrelated with time. That is, the correlation function of the noise is

$$R_N(\tau) = \sigma_N^2 e^{-|\tau|/\tau_N}$$

where τ_N is of the order of several milliseconds or less. The program approximates the noise process as discrete white noise since the correlation time is substantially smaller than the simulation time step. The discrete white noise model is

$$\delta\theta_{WN}(n+1) = \sigma_N W_R$$

where

σ_N^2 = variance of the noise

W_n = zero mean, unit variance Gaussian sample from the random number generator

(6) <u>Angular Random Walk</u>

The amplitude of a random walk process is obtained from its power spectral density or by measuring its mean square growth rate as illustrated in Figure 5.9.

The random walk process is simulated using the difference equations

$$\delta\theta_{RW}(n+1) = \delta\theta_{RW}(n) + (\sigma^2_{RW}\Delta t)^{1/2}\, W_n$$

where

σ^2_{RW}	=	amplitude of the random walk process
Δt	=	simulation time step
W_n	=	zero mean, unit variance sample from the random number generator

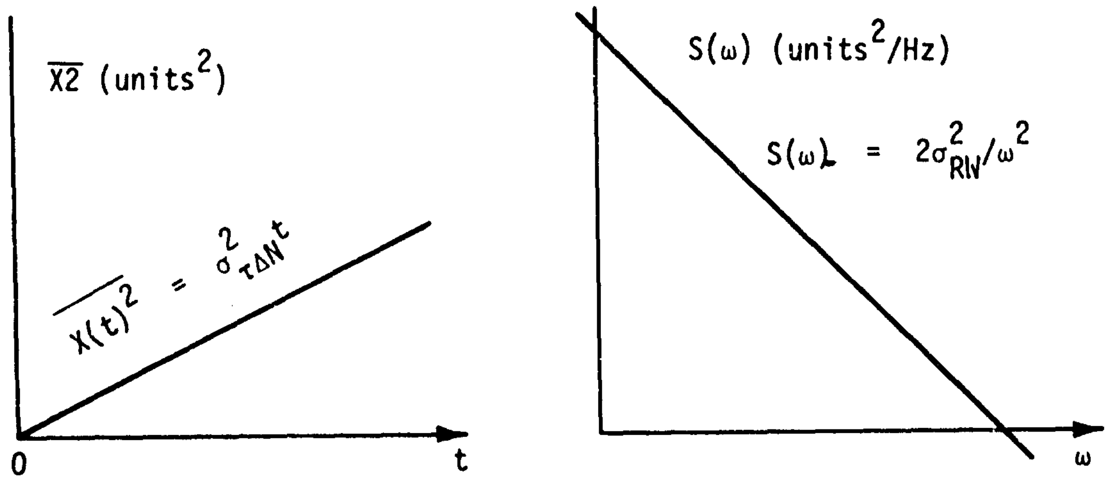


Figure 5-9. Mean square growth rate and PSD for random walk process.

(7) Measured Rotation Angle

The total measured rotation angle θ_i about the gyro input axis is

$$\underline{\omega} = (I + K_W)\underline{\omega}_{IB}^{B} + D_B + D_T$$

$$\underline{\theta}_i = \underline{\omega}\Delta t + \delta\theta_{WN} + \delta\theta_{RW}$$

where

$\underline{\omega}_{IB}^{B}$	=	angular rate of the body in body coordinates
K_W	=	gyro scale factor and IA misalignment matrix
D_B	=	gyro bias
D_T	=	turn-on drift transient
$\delta\theta_{WN}$	=	angular white noise
$\delta\theta_{RW}$	=	angular random walk
Δt	=	simulation time step

(8) Quantized Output Angle

The pulse generator computes the number of output pulses produced by the continuous angle θ_i plus the stored angle residual. The output pulses are then converted into a quantized indication of angular rotation $\Delta\theta_i$ about the input axis of the gyro.

$$\theta = \theta_i(n+1) + R(n)$$

$$n_\theta = \text{Integer } (\theta/q)$$

$$\delta\theta_i = qn_\theta$$

$$R(n+1) = \theta - qn_\theta$$

$$\Delta\theta_i(n+1) = \Delta\theta_i(n) + qn_\theta$$

where

θ_i	=	continuous indication of rotation about gyro input axis
R	=	quantized angle residual
n_θ	=	number of pulses
$\Delta\theta_i$	=	quantized indication of rotation about input axis
q	=	angular quantization

The navigation equations reset $\Delta\theta_i$ to zero each time the platform attitude matrix is computed.

5.2.3 Laser Gyro Compensation

The laser gyro compensation module (ϑCOMP) accepts incremental $\Delta\theta$ rotations from the gyros and produces compensated $\Delta\Theta$ rotations in body coordinates. The compensation module computes the compensation for the following error sources:

- Gyro input--axis misalignment
- Scale-factor error
- Scale-factor turn-on transient
- Gyro-bias drift
- Turn-on transient drift

The indicated angular rate when scale-factor errors, IA misalignments, and gyro drifts are present is

$$\underline{\omega}_i = (I + K_W)\, \omega^B_{IB} + \underline{D}_B + \underline{D}_T$$

where

K_W	=	gyro scale factor and IA misalignment matrix
K_B	=	gyro bias
D_T	=	turn-on transient drift

The estimate of platform angular rate given $\underline{W}_i$ and estimates of the error parameters are

$$\underline{\omega}_{ib}^{b} = (I + K_W)^{-1} (\underline{\omega}_i - \underline{D}_B - \underline{D}_T)$$

$$= (I - K_W) (\underline{\omega}_i - \underline{D}_B - \underline{D}_T) + O(K_W^2)$$

Since the output of the strapdown gyrus is incremental rotation angle $\Delta\Theta_i$ instead of angular rate, the compensation is

$$\delta\underline{\Theta} = \Delta\underline{\Theta}_i - (\underline{D}_B + \underline{D}_T)\Delta t$$

$$\Delta\underline{\Theta}_c = \delta\underline{\Theta} - (K_W \delta\underline{\Theta})$$

where

$\Delta\Theta_c$	=	compensated incremental rotation in body coordinates.
$\Delta\Theta_i$	=	incremental rotation obtained from the gyros.
D_B	=	gyro bias compensation.
D_T	=	turn-on transient drift compensation.
K_W	=	scale-factor and IA misalignment compensation.
Δt	=	platform attitude matrix update interval.

The off-diagonal elements contain the compensation for gyro input-axis misalignment. As in the laser gyro simulation, the elements of K_W are specified directly in the compensation module input data file rather than specifying the scale-factor values and the misalignment angles

$$K_{ii} = S_{0i} \qquad \text{(ppm)}$$

$$K_{ij} = \pm\nu_{ik} \qquad \text{(microrad)}$$

(1) Exponential Turn-on Drift Transient

A simple exponential is used to compensate the turn-on drift transient.

$$D_T = D_T(0)\ d^{-t/\tau_0}$$

where

$D_{T(0)}$ = initial compensation amplitude.

τ_0 = exponential transient time constant.

(2) Scale-Factor Turn-On Transient

The compensation for the scale-factor transient is the exponential function

$$S_T = S_{T(0)}\ e^{-t/\tau_s}$$

where

$S_{T(0)}$ = initial compensation aplitude.

τ_s = exponential transient time constant.

The elements on the main diagonal of the K_W matrix contain the total scale-factor compensation

$$K_{ii} = S_0 + S_T$$

SECTION 6

ACCELEROMETER MODELS AND COMPENSATION

The program simulates three body mounted SDF pendulous accelerometers including the change in specific force between the accelerometer locations and a specified location in the vehicle. The accelerometer compensation module contains compensation for the lever-arm effects.

6.1 Accelerometer Module

The accelerometer module (ACCEL) simulates three strapped-down, SDF, pendulous, floated accelerometers plus accelerometer lever-arm effects. The output of the module is incremental velocity in accelerometer coordinates. The SDF pendulous accelerometer contains numerous error sources in strapdown environments due to the high angular rates which are present. Only the most dominant of these error terms - anisoinertia and output - axis accelerations - are included in the accelerometer model. Float suspension, float misalignment, and products of inertia are ignored. The SDF accelerometer model contains the following error sources:

- lever-arm effects
- anisoinertia
- output-axis acceleration
- accelerometer input-axis misalignment
- scale-factor error (both positive and negative)
- scale-factor rate sensitivity (both positive and negative)
- accelerometer bias

- exponentially correlated accelerometer noise
- g and g-squared coefficients
- quantization

The program allows the user to specify one of three models for the accelerometer rebalance loop:

(a) A "performance" model which ignores accelerometer float inertia and accelerometer damping.

(b) A first-order differential equation model which contains the accelerometer damping.

(c) A second-order differential equation model which contains both accelerometer damping and inertia.

The validity of these approximations was examined previously for the SDF gyro in Section 5.1.2. The zero-order and first-order approximation artifically increase the gain of the accelerometer rebalance loop at high frequencies. The first-order approximation is reasonable because it has negligible effect on the bandwidth of the loop (see Figure 5-4). The zero-order "performance" model completely eliminates the loop dynamics, and thus produces an infinite bandwidth.

Section 6.1.1 derives the equations describing the dynamics of the SDF pendulous accelerometer. The torque-rebalance loop is then constructed using the dynamics of the instrument about its output axis. Section 6.1.2 describes the mechanization of the accelerometer module. The accelerometer compensation is discussed in Section 6.1.3.

6.1.1 Dynamics of the SDF Pendulous Accelerometer

The SDF floated pendulous accelerometer contains a pendulous mass housed in a gimbal which is floated and damped in a viscous neutral buoyancy fluid. The unbalanced float makes the instrument sensitive to linear accelerations. The motion of the float about the output axis is a direct indication of specific force along the input axis of the instrument. Thus, the instrument may be visualized as a SDF gyro with the rotor replaced with a pendulous mass. Figure 6-1 is a simplified sketch of the device. The instrument coordinates are called the input, output and pendulous (i, o, p) axes.

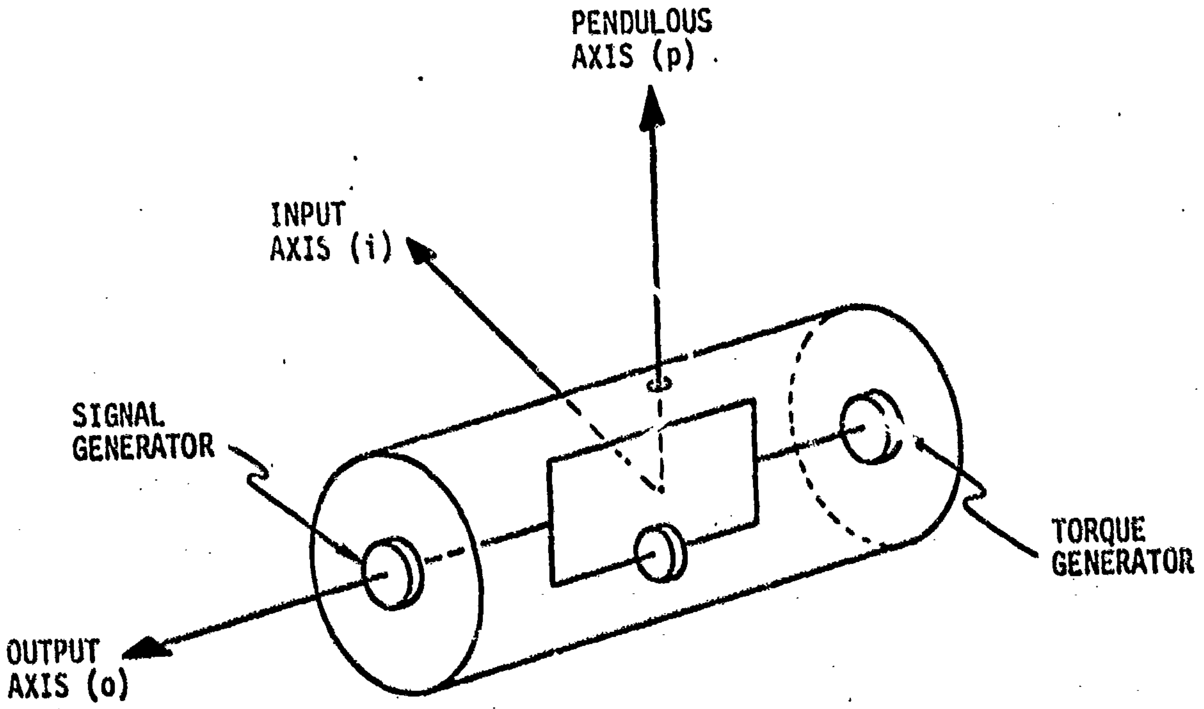


Figure 6-1. SDF pendulous accelerometer.

This section develops the dynamics of the SDF pendulous accelerometer to the extent they are represented in the accelerometer module. While the dominant characteristics of the instrument are included in the model, other known mechanisms have been emitted. The following assumptions are implicit in the derivation:

(a) The float is perfectly aligned with the accelerometer case and can rotate relative to the case only about the output axis. This implies infinite rotational suspension stiffness.

(b) The products of inertia of the float assembly (gimbal with pendulous unbalance) are zero.

The development of the accelerometer dynamics will closely follow the development in Section 5.1.1 for the single-degree-of-freedom gyro. This approach allows the two types of instruments to be readily compared. The dynamics of the accelerometer are obtained by applying Newton's second law to the float assembly. While gyros are designed to be perfectly balanced about the output axis, this is not true for the pendulous accelerometer. A known unbalance is deliberately designed into the float. The accelerometer dynamics are obtained by taking moments about point (0) in Figure 6-1. Point (0) is fixed in the float and lies at the origin of the float (i, o, p) coordinate system.

$$\underline{M}_{fo} = \left(\frac{d}{dt}\underline{H}_{fo}\right)_I + m\left(\frac{d}{dt}\underline{r}_o\right)_I \times \left(\frac{d}{dt}\underline{r}\right)_I \qquad (6\text{-}1)$$

where

$\underline{M}_{fo}$ = torque applied to the float assembly about point (0).

$\underline{H}_{fo}$ = angular momentum of the float assembly about point (0).

m = mass of the float assembly.

$\underline{r}_o$ = position of point (0).

$\underline{r}$ = position vector from point (0) to the center of mass.

$\left(\frac{d}{dt}\underline{H}_{fo}\right)_I$ = time derivative of $\underline{H}_{fo}$ wrt inertial space.

Since point (o) is fixed in the float, the angular momentum about point (o) is

$$\underline{H}_{fo} = I_o \underline{\omega}_{if} - \left(\frac{d}{dt}\underline{r}_o\right)_I \times m\underline{r} \qquad (6\text{-}2)$$

Substituting Eq. (6-2) into Eq. (6-1) gives

$$\underline{M}_{fo} = \left(\frac{d}{dt}\underline{H}_{fo}\right)_I + m\left(\frac{d}{dt}\underline{r}_o\right)_I \times \left(\frac{d}{dt}\underline{r}\right)_I$$

$$= \frac{d}{dt}\left(I_o\underline{\omega}_{if}\right)_I - \left(\frac{d^2}{dt^2}\underline{r}_o\right)_I \times m\underline{r}$$

The torque applied to the float consists of mass attraction torque plus other mechanisms.

$$\underline{M}_{fo} = \underline{M}'_{fo} + m\underline{r} \times \underline{g}_m$$

Thus, the reaction to the nongravitational torques is

$$\underline{M}'_{fo} = \frac{d}{dt}\left(I_o\underline{\omega}_{if}\right)_I - \left[\left(\frac{d^2}{dt^2}\underline{r}_o\right)_I - \underline{g}_m\right] \times m\underline{r}$$

$$= \frac{d}{dt}\left(I_o\underline{\omega}_{if}\right)_I - \underline{a} \times m\underline{r} \qquad (6\text{-}3)$$

where $\underline{a}$ is the specific force. It is more convenient to work with the time derivative with respect to the float. The Coriolis relation gives

$$\underline{M}'_{fo} = \frac{d}{dt}\left(I_o\underline{\omega}_{if}\right)_f + \underline{w}_{if} \times \left(I_o\omega_{if}\right) + m\underline{r} \times \underline{a}$$

The inertia matrix is constant because point (o) is fixed in the float assembly. Expressed in float coordinates (f),

$$\underline{M}^f_{fo} = I_o \dot{\underline{\omega}}^f_{if} + F(\underline{\omega}^f_{if})(I_o \underline{\omega}^f_{if}) + mF(\underline{r}^f)\underline{a}^f \qquad (6\text{-}4)$$

where F(.) is the matrix representation of the vector product operator.

$$F(\underline{w}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

The objective is to express the float dynamics in terms of the angular velocity and specific force in case coordinates. Expressing the angular velocity of the float as the angular velocity of the case plus the angular velocity of the float with respect to the case gives

$$\begin{aligned} \underline{\omega}^f_{if} &= \underline{\omega}^f_{ic} + \underline{\omega}^f_{cf} \\ &= C^f_c\,(\underline{\omega}^c_{ic} + \underline{\omega}^c_{cf}) \qquad (6\text{-}5) \end{aligned}$$

where

$$\underline{\omega}^c_{ic} = (\omega_i, \omega_o, \omega_p)$$

According to the initial assumptions, the float can rotate relative to the case only about the output axis; i.e.,

$$\underline{\omega}^c_{cf} = (o, \theta_o, o) \qquad (6\text{-}6)$$

and the principal axes of the float assem'‚ly are perfectly aligned with the (i, o, p) case axes when the float output angle is zero. Thus, the transformation from case to float is

$$C_c^f = \begin{bmatrix} 1 & 0 & -\theta_o \\ 0 & 1 & 0 \\ \theta_o & 0 & 1 \end{bmatrix} \tag{6-7}$$

The angular velocity of the float in float coordinates is

$$\underline{\omega}_{if}^f = \begin{bmatrix} \omega_i & - & \theta_o\omega_p \\ \omega_o & + & \dot{\theta}_o \\ \omega_p & + & \theta_o\omega_i \end{bmatrix} \tag{6-8}$$

The specific force in float coordinates is

$$\underline{a}^f = C_c^f \, \underline{a}^c = \begin{bmatrix} a_i - \theta_o a_p \\ a_o \\ a_p + \theta_o a_i \end{bmatrix} \tag{6-9}$$

Substituting Eq. (6-8) and Eq. (6-9) into

$$\underline{M}_{fo}^f = I_o \dot{\underline{\omega}}_{if}^f + F(\underline{\omega}_{if}^f)(I_o \underline{\omega}_{if}^f) + mF)\underline{r}^f)\underline{a}^f \tag{6-4}$$

gives the following expression for the dynamics about the output axis.

$$M_o = I_o\ddot{\theta}_o + (I_i - I_p)\omega_i\omega_p + \theta_o(I_i - I_p)(\omega_i^2 - \omega_p^2)$$
$$+ I_o\dot{\omega}_o - mr_p a_i + \theta_o mr_p a_p \qquad (6\text{-}10)$$

The output axis equation was derived assuming the products of inertia are zero; i.e.,

$$I_o = \begin{bmatrix} I_i & 0 & 0 \\ 0 & I_o & 0 \\ 0 & 0 & I_p \end{bmatrix}$$

and the pendulous mass lies along the negative (p) axis

$$\underline{r}^f = \{o, o, -r_p\}$$

The product mr_p is called the pendulosity.

$$P = mr_p$$

The torque applied about the output axis, M_o, is assumed to consist of

(a) Torque from the torque generator, M_{tg}.

(b) Viscous torque opposing the output axis rotation, $C_o\dot{\theta}_o$.

(c) Gyro disturbance torques, M_d.

That is,

$$M_o = M_{tg} - C_o\dot{\theta}_o + M_d \qquad (6\text{-}11)$$

Substituting Eq. (6-11) into Eq. (6-10) and rearranging terms gives

$$\begin{aligned}
I_o\ddot{\theta}_o + C_o\dot{\theta}_o = {} & Pa_i + M_{tg} && \text{(ideal accelerometer)} \\
& + (I_p - I_i)\omega_i\omega_p && \text{(anisoinertia torque)} \\
& + \theta_o(I_p - I_i)(\omega_i^2 - \omega_p^2) && \text{(anisoinertia coupling)} \\
& - I_o\dot{\omega}_o && \text{(output-axis angular acceleration)} \\
& - \theta_o Pa_p && \text{(cross-coupling torque)} \\
& + M_d && \text{(disturbance torques)}
\end{aligned} \qquad (6\text{-}12)$$

The accelerometer rebalance loop is designed using the ideal accelerometer equation

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o = Pa_i + M_{tg}$$

Since the other terms are ignored, they produce errors in the indicated specific force and must be compensated in the navigation computer. Current inertial navigation systems use digital accelerometer rebalance loops. A typical digital loop is shown in Figure 6-2. Each output pulse represents an increment of velocity (integral of specific force) along the input axis. All the error torques have been lumped into M_d for convenience.

The rebalance loop mechanized in the accelerometer module is shown in Figure 6-3. The rebalance torque is generated by multiplying the float output angle by the linear gain K where

$$K = K_{sg} K_{tg}$$

Thus, the program simulates an analog rebalance loop followed by a delta-modulator to generate the incremental velocity output.

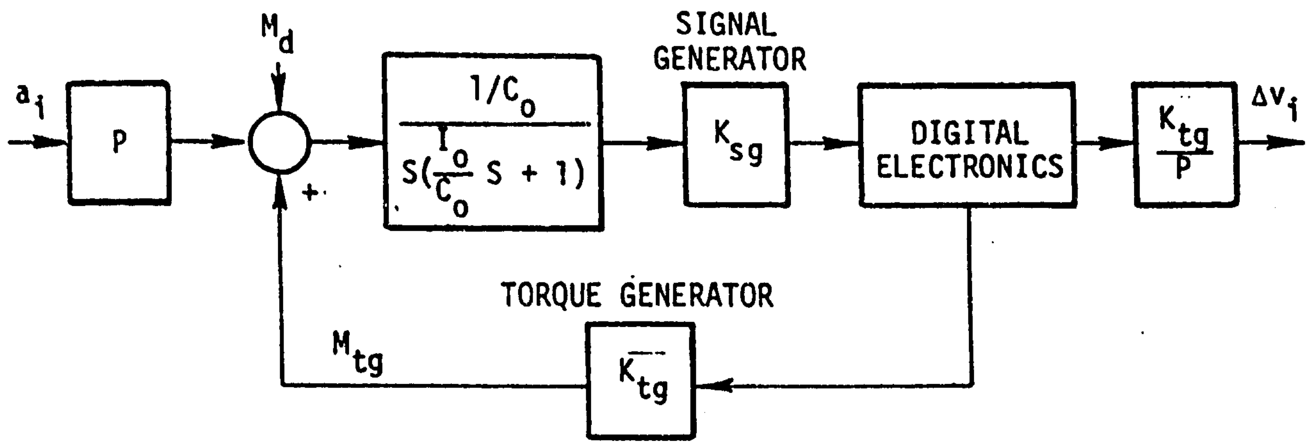
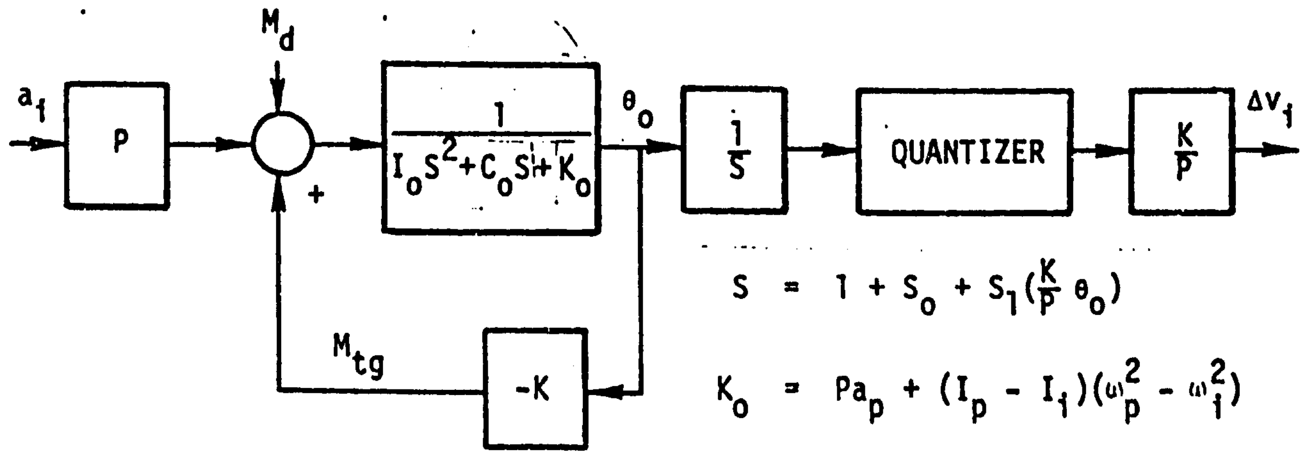


Figure 6-2. Pulse-torquing accelerometer rebalance loop.



$$S = 1 + S_o + S_1(\frac{K}{P}\theta_o)$$

$$K_o = Pa_p + (I_p - I_i)(\omega_p^2 - \omega_i^2)$$

Figure 6-3. Simulated analog accelerometer rebalance loop.

The rebalance loop equations are obtained by substituting

$$M_{tg} = -K\theta_o$$

into Eq. (6-12) and rearranging terms.

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + [Pa_p + (I_p - I_i)(\omega_p^2 - \omega_i^2)$$

$$+ K]\theta_o = Pa_i + M_d + (I_p - I_i)\omega_i\omega_p - I_o\dot{\omega}_o$$

Again, it is convenient to lump all of the sensor errors into the disturbance torque M_d. This gives

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + K_1\theta_o = Pa_i + M_d'$$

$$K_1 = Pa_p + (I_p - I_i)(\omega_p^2 - \omega_i^2) + K$$

$$M_d' = M_d + (I_p - I_i)\omega_i\omega_p - I_o\dot{\omega}_o$$

(6-13)

The accelerometer module allows the user to selectively solve for θ_o in Eq. (6-13) using one of three models for the rebalance loop:

(a) A "performance" model which ignores float inertia and damping; i.e.,

$$\theta_o = (Pa_i + M_d')/K_1$$

(b) A first-order differential equation which includes the accelerometer damping; i.e.,

$$C_o\dot{\theta}_o + K_1\theta_o = Pa_i + M_d'$$

(c) The complete second-order model; i.e.,

$$I_o\ddot{\theta}_o + C_o\dot{\theta}_o + K_1\,{}_o = Pa_i + M_d'$$

The effect of using the zero-order or first-order approximations has been discussed, but to reiterate - while the first-order model is reasonable the "performance" model produces an infinite rebalance loop bandwidth.

6.1.2 Accelerometer Module Equations

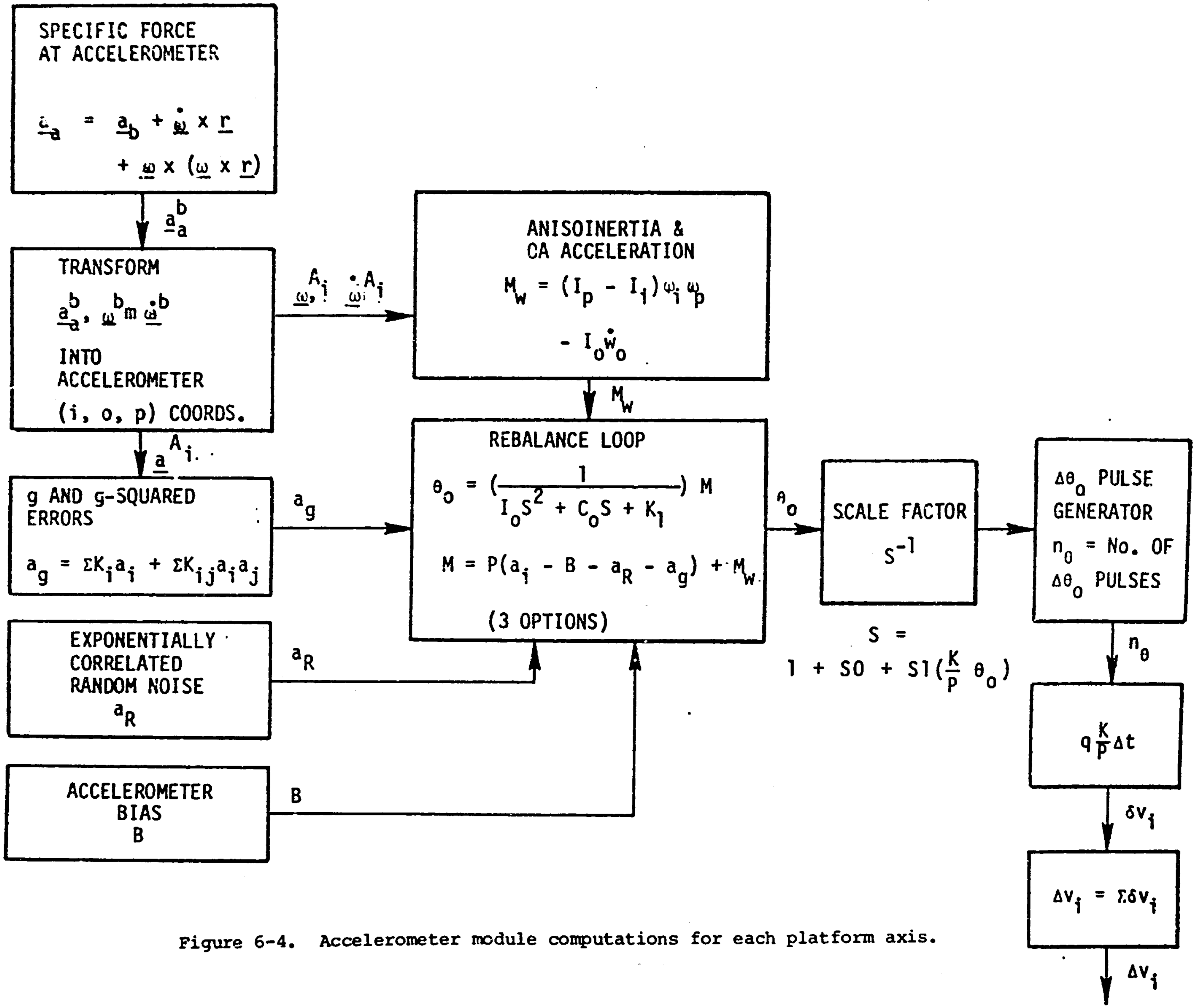
The accelerometer module simulates the three accelerometer rebalance loops sequentially. Figure 6-4 shows the computations for one platform axis. The computations for each of the axes are identical. The equations indicated in Figure 6-4 are summarized in this section.

(1) Lever-Arm Effects

The displacement of each accelerometer from the vehicle reference location is specified in the accelerometer module input data file. The lever arm is specified in body coordinates. The displacement of the accelerometer from the reference location changes the output of the instrument. The change in specific force can be derived from Figure 6-5.

The specific force at the accelerometer location expressed in body coordinates is

$$\begin{aligned}\underline{a}_a^b &= \underline{a}_b^b + \left(\frac{d^2\,\underline{r}^b}{dt^2}\right)_I \\ &= \underline{a}_b^b + \underline{\omega}_{ib}^b \times (\underline{\omega}_{ib}^b \times \underline{r}^b) + \dot{\underline{\omega}}_{ib}^b \times \underline{r}^b \\ &\quad + \dot{\underline{r}}^b + 2\underline{\omega}_{ib}^b \times \dot{\underline{r}}^b\end{aligned}$$



SPECIFIC FORCE AT ACCELEROMETER
$\underline{a}_a = \underline{a}_b + \dot{\underline{\omega}} \times \underline{r} + \underline{\omega} \times (\underline{\omega} \times \underline{r})$
$\underline{a}_a^b$
TRANSFORM $\underline{a}_a^b$, $\underline{\omega}^b$ m $\dot{\underline{\omega}}^b$ INTO ACCELEROMETER (i, o, p) COORDS.
$\underline{\omega}^{A}_{i}$, $\dot{\underline{\omega}}^{A}_{i}$
ANISOINERTIA & CA ACCELERATION
$M_W = (I_p - I_i)\omega_i\omega_p - I_o\dot{w}_o$
M_W
$\underline{a}^{A}_{i}$
g AND g-SQUARED ERRORS
$a_g = \Sigma K_i a_i + \Sigma K_{ij} a_i a_j$
a_g
REBALANCE LOOP
$\theta_o = (\frac{1}{I_o S^2 + C_o S + K_1}) M$
$M = P(a_i - B - a_R - a_g) + M_W$
(3 OPTIONS)
EXPONENTIALLY CORRELATED RANDOM NOISE a_R
a_R
ACCELEROMETER BIAS B
B
θ_o
SCALE FACTOR S^{-1}
$S = 1 + S0 + S1(\frac{K}{P}\theta_o)$
$\Delta\theta_o$ PULSE GENERATOR
n_θ = No. OF $\Delta\theta_o$ PULSES
n_θ
$q\frac{K}{P}\Delta t$
δv_i
$\Delta v_i = \Sigma\delta v_i$
Δv_i

Figure 6-4. Accelerometer module computations for each platform axis.

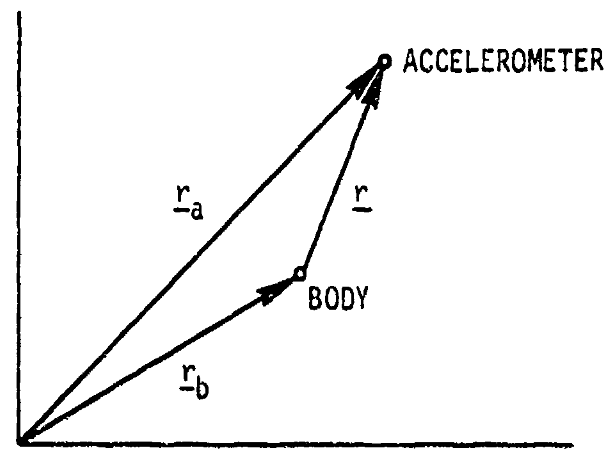


Figure 6-5. Geometry of the lever-arm effect.

The first and second derivatives of $\underline{r}$ are zero because the lever arm is assumed to be fixed in the body. Thus, for a rigid lever arm, the specific force at the accelerometer location is equal to the specific force at the reference location plus the centrepital and tangential acceleration produced by vehicle angular motion.

$$\underline{a}_a^b = \underline{a}_b^b + \underline{\omega}_{ib}^b \times (\underline{\omega}_{ib}^b \times \underline{r}^b) + \dot{\underline{\omega}}_{ib}^b \times \underline{r}^b \qquad (6\text{-}14)$$

The program mechanizes Eq. (6-14).

(2) Transformation into Accelerometer Coordinates

The specific force, angular velocity, and derivative of angular velocity generated by the random motion module (ENV) are transformed into accelerometer (i, o, p) coordinates.

$$\underline{a}^{A_i} = C_b^{A_i}\,\underline{a}^b$$

$$\underline{\omega}^{A_i} = C_b^{A_i}\,\underline{\omega}^b \qquad (i = 1, 2, 3)$$

$$\dot{\underline{\omega}}^{A_i} = C_b^{A_i}\,\dot{\underline{\omega}}^b$$

where $C_b^{A_i}$ is the transformation to (i, o, p) coordinates of the ith accelerometer. The elements of $C_b^{A_i}$ are not programmed constants—they are specified by the accelerometer initialization data set. The accelerometer input-axis misalignment must be incorporated into the $C_b^{A_i}$ input data.

(3) g and g-Squared Coefficients

The accelerometer error generated by the g and g-squared coefficients is obtained by multiplying each coefficient by the appropriate component of $\underline{a}^{A_i}$

$$\underline{a}_g = K_o a_o + K_p a_p + K_{ii} a_i^2 + K_{io} a_i a_o$$
$$K_{ip} a_i a_p + K_{op} a_o a_p + K_{pp} a_p^2$$

where

$\underline{a}^{A_i}$ = (a_i, a_o, a_p).

K_k = linear acceleration sensitivity for acceleration along the kth accelerometer axis.

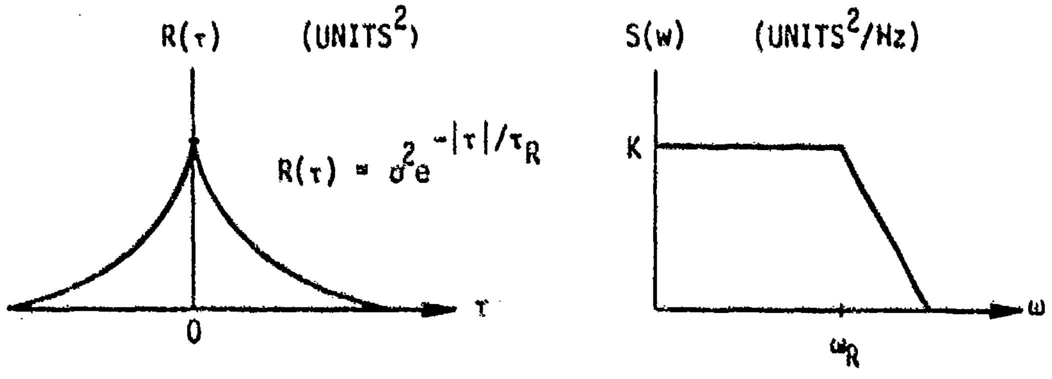
K_{jk} = g-squared sensitivity (compliance) for acceleration along the j and k accelerometer axes.

(4) Accelerometer Bias

The accelerometer bias B represents the constant disturbance torque applied to the accelerometer float. Typically, there is some shift in this coefficient each time the system is turned on.

(5) Exponentially Correlated Random Noise

Exponentially correlated random accelerometer noise is used to describe the stability of the following turn-on. The amplitude and correlation time are obtained from power spectral density plots of instrument test data. The two parameters are uniquely defined by the amplitude and break frequency of the PSD. Figure 6-6 shows the required relations. A more detailed discussion of an exponentially correlated random process is presented in Section 5.1.3(4).



$$\boxed{\begin{aligned} \sigma^2 &= K\omega_R \\ \tau_R &= 1/\omega_R \end{aligned}}$$

Figure 6-6. Autocorrelation function and PSD for exponentially correlated random noise.

The accelerometer module simulates an exponentially correlated time series using

$$a_R(n + 1) = a_R(n)e^{-\Delta t/\tau_R} + \sqrt{\sigma^2\left(1 - e^{-2\Delta t/\tau_R}\right)}w_n;\ a_R(o) = 0$$

where

a_R	=	simulated random accelerometer noise.
Δt	=	simulation time step.
τ_R	=	correlation time.
σ^2	=	steady-state variance of the random process.
w_n	=	zero-mean, unit-variance gaussian sample from the random-number generator.

Since the random process is initialized to zero, the mean square value approaches the steady-state variance in an exponential manner.

$$\sigma^2_{a_R}(t) = \sigma^2\left(1 - e^{-2t/\tau_R}\right)$$

(6) Anisoinertia and OA Acceleration Torques

The sum of the anisoinertia and output-axis acceleration torques is

$$M_w = (I_p - I_i)\ \omega_i\omega_p - I_o\dot{\omega}_o$$

where

I_i, I_o, I_p	=	principal inertias of the float about (i, o, p).
ω_i, ω_p	=	angular velocity of the case about the input and spin axes.
$\dot{\omega}_o$	=	time derivative of the angular velocity of the case about the output axis.

(7) Rebalance Loop Dynamics

The float output angle θ_o may be computed using one of the three options shown below.

$$\underbrace{I_o\ddot{\theta}_o + \underbrace{C_o\dot{\theta}_o + \underbrace{K_1\theta_o = M}_{\text{PERFORMANCE MODEL}}}_{\text{FIRST-ORDER MODEL}}}_{\text{SECOND-ORDER MODEL}}$$

$$K_1 = Pa_p + (I_p - I_i)(\omega_p^2 - \omega_i^2) + K$$

$$M = P(a_i - B - a_R - a_{\underline{g}}) + M_w$$

where

B	=	accelerometer bias.
a_R	=	exponentially correlated accelerometer noise.
$a_{\underline{g}}$	=	g and g-squared sensitive error.
M_w	=	anisoinertia and output-axis acceleration torques.

This is a direct mechanization of the rebalance loop dynamics shown in Eq. (6-13). The impact of approximating the rebalance loop with either the "performance" or first-order models is discussed in Section 6.1. The discrete equations for each option are:

(a) Performance Model

$$\theta_o(n + 1) = M(n + 1)/K_1$$

(b) First-Order Model

$$\theta_o(n + 1) = \theta_o(n) + \frac{1}{C_o}[-K_1\theta_o(n) + M(n + 1)]\Delta t$$

$$\theta_o(o) = Pa_i(o)/K$$

(c) Second-Order Model

$$\theta_o(n + 1) = \theta_o(n) + \dot{\theta}_o(n)\ T$$

$$\dot{\theta}_o(n + 1) = \dot{\theta}_o(n) + \frac{1}{I_o}[-C_o\dot{\theta}_o(n) - K_1\theta_o(n) + M(n + 1)]\Delta t$$

$$\dot{\theta}_o(o) = 0$$

$$\theta_o(o) = Pa_i(o)/K$$

The program defines accelerometer bias, random accelerometer-noise, and the [illegible] and g-squared errors with the opposite sign from the normal convention.

(8) Scale Factor

The program uses the following definition for accelerometer scale factor:

$$\tilde{a}_i = s^{-1} a_i$$

where a_i is the actual specific force and $\tilde{a}_i$ is the accelerometer output; i.e., an increase in accelerometer scale factor decreases the magnitude of the accelerometer output. The scale factor consists of a linear term plus a term proportional to the specific force.

$$S = [1 + S0 + S1a_i]$$

$$= [1 + S0 + S1\ (\frac{K}{P}\theta_o)]$$

The scale-factor coefficients may have different values for positive and negative values of specific force. The scale-factor equations are:

$$S = 1 + S0_{(+)} + S1_{(+)}(\frac{K}{P}\theta_o) \quad \theta_o \geq 0$$

$$S = 1 + S0_{(-)} + S1_{(-)}(\frac{K}{P}\theta_o) \quad \theta_o < 0$$

where

$S0_{(+)}, S0_{(-)}$ = positive and negative linear scale factors.

$S1_{(+)}, S1_{(-)}$ = positive and negative rate sensitive scale factors.

(9) Velocity Quantization

The pulse generator computes the number of output pulses produced by the output angle θ_o plus the stored angle residual.

$$\theta = \theta_o(n + 1) + \theta_R(n) \qquad \theta_R(o) = 0$$

If $(\theta \geq 0)$ $S = 1 + S0_{(+)} + S1_{(+)}(\frac{K}{P}\theta)$

If $(\theta \leq 0)$ $S = 1 + S0_{(-)} + S1_{(-)}(\frac{K}{P}\theta)$

$$n_\theta = \text{Integer}\ (\frac{\theta}{qs})$$

$$\tilde{\theta}_o = qSn$$

$$\theta_R(n + 1) = \theta - \tilde{\theta}_o$$

where

$\tilde{\theta}_o$ = quantized indication of float output angle.

θ_o = float output angle.

θ_R = quantized angle residual.

S = scale factor.

q = $\Delta\theta_o$ quantization level.

n_θ = number of $\Delta\theta_o$ pulses.

The output pulses are then converted into an indication of the change in velocity (integral of specific force) along the input axis of the accelerometer. The output angle is a direct indication of specific force.

$$\begin{aligned} a_i &= \frac{K}{P} S^{-1} \tilde{\theta}_o \\ &= \frac{K}{P} q n_\theta \end{aligned}$$

Thus, the integral of the specific force along the accelerometer input axis over the simulation time step is

$$\begin{aligned} \delta v_i &= \tilde{a}_i \Delta t \\ &= \frac{K}{P} q n_\theta \Delta t \end{aligned}$$

The accumulated incremental velocity is

$$\Delta v_i = \Sigma \delta v_i$$

The navigation equations reset Δv_i to zero each time the incremental velocity is transformed through the platform attitude matrix.

6.1.3 Accelerometer Compensation

The accelerometer compensation module (ACOMP) accepts incremental velocity from the simulated SDF pendulous accelerometers and produces compensated incremental velocity in body coordinates. The following errors are compensated:

- Accelerometer input-axis misalignment
- Scale-factor error (positive and negative).
- Scale-factor acceleration sensitivity (positive and negative.
- Accelerometer bias.
- K_{ii} g-squared coefficients.
- Anisoinertia torque.
- Output-axis acceleration torque.
- Lever-arm effects.

A flow diagram of the compensation module is shown in Figure 6-7. The compensation equations for each platform axis are summarized below.

(1) Scale-Factor Compensation

The incremental velocities from each accelerometer are compensated for linear and acceleration sensitive scale factors. The accelerometer output expressed at the acceleration level is

$$\tilde{a}_i = S^{-1}(a_i + \delta a)$$

where a_i is the actual specific force and δa is the additive accelerometer error. Thus, the estimated specific force given a_i and estimates of the scale-factor and accelerometer error is

$$a_i = S\tilde{a}_i - \delta a$$

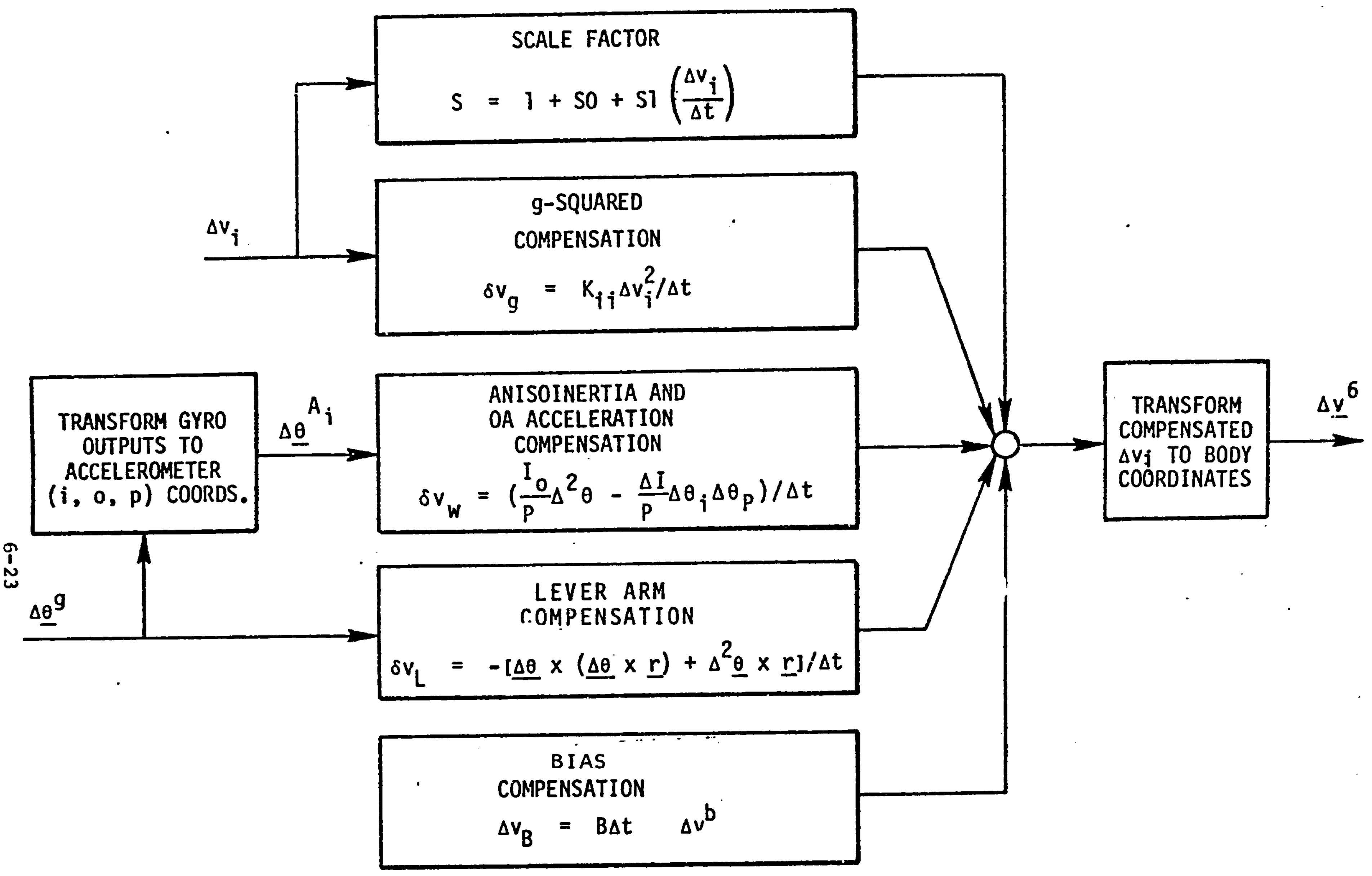

SCALE FACTOR
$S = 1 + S0 + S1\left(\frac{\Delta v_i}{\Delta t}\right)$
g-SQUARED COMPENSATION
$\delta v_g = K_{ii}\Delta v_i^2/\Delta t$
ANISOINERTIA AND OA ACCELERATION COMPENSATION
$\delta v_w = (\frac{I_o}{P}\Delta^2\theta - \frac{\Delta I}{P}\Delta\theta_i\Delta\theta_p)/\Delta t$
LEVER ARM COMPENSATION
$\delta v_L = -[\underline{\Delta\theta} \times (\underline{\Delta\theta} \times \underline{r}) + \Delta^2\underline{\theta} \times \underline{r}]/\Delta t$
BIAS COMPENSATION
$\Delta v_B = B\Delta t \quad \Delta v^b$
TRANSFORM GYRO OUTPUTS TO ACCELEROMETER (i, o, p) COORDS.
TRANSFORM COMPENSATED Δv_i TO BODY COORDINATES
Δv_i
$\underline{\Delta\theta}^{A_i}$
$\underline{\Delta\theta}^g$
$\Delta\underline{v}^6$

Figure 6-7. Accelerometer compensation for each platform axis.

Since the output of the accelerometer is incremental velocity rather than specific force, the compensation is

$$\Delta v_i = S\Delta\tilde{v}_i - \delta a\Delta t$$

Separate scale factors for positive and negative specific force are provided in the compensation module.

$$S = 1 + S0_{(+)} + S1_{(+)}\left(\frac{\Delta\tilde{v}_i}{\Delta t}\right), \ \Delta\tilde{v} \geq 0$$

$$S = 1 + S0_{(-)} + S1_{(-)}\left(\frac{\Delta\tilde{v}_i}{\Delta t}\right), \ \Delta\tilde{v}_i < 0$$

where

$S0_{(+)}$, $S0_{(-)}$	=	positive and negative linear scale factors.
$S1_{(+)}$, $S1_{(-)}$	=	positive and negative acceleration-sensitive scale factors.

(2) <u>g-Squared Sensitive Error Compensation</u>

Compensation is provided for the K_{ii} coefficients, i.e., the terms which are sensitive to the square of acceleration along the input axis. The g-sensitive K_o and K_p coefficients are not compensated nor are the K_{io}, K_{is}, K_{ss} g-squared terms. The acceleration error produced by K_{ii} is

$$\delta a_i = -K_{ii}a_i^2$$

The g-squared compensation is

$$\delta v_g = K_{ii}\Delta v_i^2/\Delta t$$

where K_{ii} is the compensation coefficient and $1/\Delta t$ is the rate at which the incremental velocities are transformed through the platform attitude matrix.

(3) Acceleration Bias Compensation

Acceleration bias produces a constant acceleration error.

$$\delta a_i = -B$$

The bias compensation is

$$\delta v_i = B\Delta t$$

(4) Anisoinertia Torque Compensation

Anisoinertia torque produces the acceleration error

$$\delta a_i = \frac{1}{P}(I_p - I_i)\omega_i\omega_p$$

where

P = accelerometer pendulosity.

I_i, I_i = principal inertias of the float about (i, p).

The compensation is

$$\delta v_i = -\frac{1}{P}(I_p - I_i)\Delta\theta_i\Delta\theta_p/\Delta t$$

where $\Delta\theta_i$ and $\Delta\theta_p$ are incremental rotations about the accelerometer input and pendulous axes.

(5) Output-Axis Acceleration Torque Compensation

Output-axis acceleration produces the acceleration error

$$\delta a_i = -\frac{I_o}{P}\dot{\omega}_o$$

where

P = accelerometer pendulosity.

I_o = principal inertia about the float output axis.

The compensation is

$$\delta v_i = \frac{I_o}{P}[\Delta\theta_o(n) - {}'\Delta\theta_o(n-1)]/\Delta t$$

where $\Delta\theta_o$ is the incremental rotation about the output axis.

(6) Lever-Arm Compensation

For a rigid lever arm, the specific force at the location of the accelerometer is equal to the specific force at the reference position plus the centripal and tangential acceleration produced by vehicle angular motion.

$$a_a^b = a_b^b + \underline{\omega}_{ib}^b \times (\underline{\omega}_{ib}^b \times \underline{r}^b) + \dot{\underline{\omega}}_{ib}^b \times \underline{r}^b$$

where

a_a^b = specific force at the accelerometer in body coordinates.

a_b^b = specific force at the cg in body coordinates.

$\underline{r}^b$ = lever arm in body coordinates.

The objective is to compute the specific force at the vehicle center of gravity. The lever-arm correction for the i th accelerometer is obtained by evaluating the i th component of

$$\delta \underline{v}_L^a = -[\Delta\underline{\theta}^g \times (\Delta\underline{\theta}^g \times \underline{r}^b) + \Delta^2\underline{\theta}^b \times \underline{r}^b]/\Delta t$$

where

$\Delta^2\underline{\theta}^g = \Delta\underline{\theta}^g(n) - \Delta\underline{\theta}^g(n-1)$

$\Delta\theta^g$ = incremental rotations from the simulated gyros

The gyro outputs $\Delta\underline{\theta}^g$ are the values before gyro compensation.

(7) <u>Total Accelerometer Compensation</u>

The compensated incremental velocity along the accelerometer input axis is

$$\Delta v_i(\text{comp}) = S(\Delta v_i) + (\delta v_B + \delta v_g + \delta v_I + \delta v_{OA} + \delta v_L)$$

where

S = scale-factor compensation.

δv_B = bias compensation.

δv_g = K_{ii} g-squared compensation.

δv_I = anisoinertia torque compensation.

δv_{OA} = OA acceleration.

δv_L = lever-arm compensation.

The three compensated accelerometer outputs are then transformed from the accelerometer input axes to body coordinates.

$$\Delta \underline{v}^b = C_a^b \Delta \underline{v}^a$$

where

$\Delta \underline{v}^a$	=	compensated accelerometer outputs along the three accelerometer input axes.
C_a^b	=	(nonorthogonal) transformation from the accelerometer input axes to body coordinates.

The transformation matrix C_a^b is labeled QMIS in the program. The elements of this matrix must be consistent with the rotation matrices which specify the (i, o, p) axes of each accelerometer relative to body coordinates. These rotation matrices are labeled QABX, QABY, QABZ. All of the matrices are obtained from the initialization data set for the compensation module.

SECTION 7

VELOCITY-ATTITUDE ALGORITHM FOR A STRAPDOWN INERTIAL NAVIGATOR

The velocity-attitude algorithm is the second or middle module of the three main modules which comprise the navigation (and attitude) software for a SD INS. The inputs to the velocity-attitude (V-A) module are ideally the "incremental velocities" of the "center" of the vehicle in the body frame over a computation cycle, $\overline{\Delta V}^b(n)$, and the "incremental angles" through which the body has turned with respect to the inertial frame, expressed in the body frame, over a fast cycle, $\overline{\Delta\theta}^b_{ib}(n)$. The inputs from the inertial sensors are the raw pulse counts which are corrected in the preceding compensation modules to give the computer's best estimate of $\overline{\Delta V}^b(n)$ and $\overline{\Delta\theta}^b_{ib}(n)$.

The functions of the VA module in the INSS are threefold:

a. The initial value of the transformation from the body frame to the inertial frame, $C^i_b(0)$, is computed using the "loaded" values of roll, pitch, and yaw, to form C^c_b, and the "loaded" values of wander angle, latitude, and longitude, to form C^i_c. (The "loaded values of the above parameters may include initial errors in each.) In normal system operation $C^i_b(0)$ would be computed by the initial alignment routine.

b. The strapdown alignment matrix, $C^i_b(t)$, is updated every computation cycle on the basis of the incremental angles, $\overline{\Delta\theta}^b_{ib}$, from the gyros. The method may involve either a d.c.m. or quaternion update (of the first, second, or third order) but in any case the updated transformation is expressed as a d.c.m. for use in the $\overline{\Delta V}$ transformation and for attitude computation.

c. The incremental velocity, $\overline{\Delta V}^b$, is transformed from the body frame to the inertial frame every computation cycle, by premultiplying by C_b^i. In an earlier version of the program, the direction cosine matrix per se was not generated - since attitude was not computed - and the transformation was accomplished using the rotation quaternion, q_b^i, and its conjugate, q_b^{i*}, viz:

$$\overline{\Delta V}^i = q_b^i \overline{\Delta V}^b q_b^{i*} = C_b^i \overline{\Delta V}^b \tag{7-1}$$

One additional operation is performed in the V-A algorithm, simply for convenience in computing attitude; this is the generation of the transformation from the inertial (i) frame to the "new" earth fixed (e') frame, $C_i^{e'}$. This transformation accounts for the multiply defined inertial frames, i and i', and permits the direct use of the earth-fixed to "new" LVWA transformation, $C_{e'}^{c'}$, in the LVWA navigation algorithm to complete the computation of the "attitude" matrix.

7.1 Initial Strapdown "Alignment" Matrix, $C_b^i(0)$, and Initial Alignment Quaternion, $q_b^i(0)$

The initial body-to-inertial transformation is formed as the product of two rotation matrices, C_b^c and C_c^i. In the notation of Appendix B, C_b^c is

$$C_b^c = \tilde{Z}(\pi/2)X(\pi)\tilde{Z}(\psi_w)\tilde{Y}(\theta)\tilde{X}(\phi)X(\pi)$$

$$= \left[\begin{array}{c|c|c} CP\cdot SY & -SR\cdot SP\cdot SY - CR\cdot CY & -CR\cdot SP\cdot SY + SR\cdot CY \\ CP\cdot CY & -SR\cdot SP\cdot CY + CP\cdot SY & -CR\cdot SP\cdot CY - SR\cdot SY \\ SP & SR\cdot CP & CR\cdot CP \end{array}\right] \tag{7-2}$$

where

$$\left.\begin{array}{lll} R & = \phi & = \text{roll} \\ P & = \theta & = \text{pitch} \\ Y & = \psi_w & = \text{yaw} \end{array}\right\} \begin{array}{l}\text{loaded} \\ \text{values may} \\ \text{include errors}\end{array}$$

and $s = \sin$ and $c = \cos$

In the V-A module, C_b^c is denoted by QPB. (A lot of frames are the P-frame in the actual INSS program.)

C_c^i at t = 0, and hence $\omega_{ie}t = 0$, is

$$C_c^i(0) = \tilde{Y}(\ell)X(L)\tilde{Z}(\alpha) = C_c^e(0)$$

$$= \left[\begin{array}{c|c|c} CLO\cdot CW - SLO\cdot SLA\cdot SW & -CLO\cdot SW - SLO\cdot SLA\cdot CW & SLO\cdot CLA \\ CLA\cdot SW & CLA\cdot CW & SLA \\ -SLO\cdot CW - CLO\cdot SLA\cdot SW & SLO\cdot SW - CLO\cdot SLA\cdot CW & CLO\cdot CLA \end{array}\right] \tag{7-3}$$

where

$$\left.\begin{array}{lll} LO & = \ell & = \text{longitude} \\ LA & = L & = \text{latitude} \\ W & = \alpha & = \text{wander angle} \end{array}\right\} \begin{array}{l}\text{loaded} \\ \text{values may} \\ \text{include errors}\end{array}$$

In the V-A module, $C_c^i(0)$ is denoted by QIP. Finally, the body-to-inertial transformation, $C_b^i(0)$, is formed by multiplying the above two rotation matrices, i.e.,

$$C_b^i(0) = C_c^i(0)C_b^c(0) \tag{7-4}$$

This product is denoted by QIB in the V-A module

In the case where a quaternion update algorithm is to be employed, it is necessary to convert the initial body-to-inertial d.c.m. (or rotation matrix), $C_b^i(0)$, to the corresponding unit (or rotation) quaternion,

$q_b^i(0)$. Later (in the V-A module), when a quaternion update is performed, the updated, normalized (unit) quaternion is converted to a rotation matrix for $\overline{\Delta V}$-transformation and attitude computation. The link between a unit or rotation quaternion and the corresponding rotation matrix is the interpretation of the rotation matrix in terms of the direction cosines of the axis of rotation, and the angle of rotation developed in Appendix C. Calling the initial alignment matrix, $R = [r_{ij}]$, the initial (t = 0) quaternion elements are computed as indicated below:

$$
\begin{aligned}
q_0(0) &= \sqrt{\frac{1 + r_{11}(0) + r_{22}(0) + r_{33}(0)}{4}} \\
q_1(0) &= [r_{32}(0) - r_{23}(0)]/4q_0(0) \\
q_2(0) &= [r_{13}(0) - r_{31}(0)]/4q_0(0) \\
q_3(0) &= [r_{21}(0) - r_{12}(0)]/4q_0(0) \qquad (7\text{-}5)
\end{aligned}
$$

7.2 Updating the SD Alignment Matrix or Quaternion

Updating $C_b^i(t)$ or $q_b^i(t)$ from t to t + Δt, requires the integration of one or the other of the following equations:

$$\dot{C}_b^i = C_b^i \underline{\omega}_{ib}^b \quad \text{for a d.c.m. update} \qquad (7\text{-}6)$$

or

$$\dot{q}_b^i = \frac{1}{2} q_b^i \underline{\omega}_{ib}^b \quad \text{for a quaternion update} \qquad (7\text{-}7)$$

where the superscript dot denotes differentiation with respect to time.

Obviously the simplest form of integration which may be employed (first order) is to multiply the derivations by Δt and add them to the old values, viz:

7.2.1 First-Order DCM Update

$$C_b^i(t + \Delta t) = C_b^i(t) + \dot{C}_b^i(t)\Delta t$$

$$= C_b^i[I + \Omega_{ib}^b(t)\Delta t] \qquad (7\text{-}8)$$

where

$$\Omega_{ib}^b \overset{\Delta}{=} \begin{bmatrix} 0 & -\omega_{ib_3}^b & \omega_{ib_2}^b \\ \omega_{ib_2}^b & 0 & -\omega_{ib_1}^b \\ -\omega_{ib_2}^b & \omega_{ib_1}^b & 0 \end{bmatrix} \leftrightarrow \begin{bmatrix} \omega_{ib_1}^b \\ \omega_{ib_2}^b \\ \omega_{ib_3}^b \end{bmatrix} = \underline{\omega}_{ib}^b$$

If the elements of $C_b^i(t)$ are called C_{ij}: i, j = 1, 2, 3 and the non-zero elements of $\Omega_{ib}^b(t)\Delta t$ are denoted by

$$\left.\begin{aligned} \omega_{ib_1}^b \Delta t &= \Delta\theta_1 \\ \omega_{ib_2}^b \Delta &= \Delta\theta_2 \\ \omega_{ib_3}^b \Delta &= \Delta\theta_3 \end{aligned}\right\} \text{RHS's are the incremental angles from gyro compensation}$$

Then the $C_b^i(t + \Delta t)$ resulting from the first-order update is

$$C_b^i(t + \Delta t) \approx \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} 1 & -\Delta\theta_2 & \\ \Delta\theta & 1 & -\Delta\theta_1 \\ -\Delta\theta_2 & \Delta\theta_1 & 1 \end{bmatrix} \qquad (7\text{-}8a)$$

Unless all the $\Delta\theta_i$ are zero, $C_b^i(t + \Delta t)$ will not be a true rotation matrix. The criterion for a rotation matrix is

$$C\tilde{C} = CC^{-1} = \tilde{C}C = C^{-1}C = I \qquad (7\text{-}9)$$

or

$$\det C = \det \tilde{C} = \det (C\tilde{C}) = +1$$

To attain (or approach) the above condition the updated d.c.m. is (at least periodically) "orthormalized" - to the first order - by using the following algorithm:

$$C_{orth} = C - \frac{1}{2} C(\tilde{C}C - I) \qquad (7\text{-}10)$$

7.2.2 First-Order Quaternion Update

By analogy with Eq. (7-8), we may write the first order quaternion update as

$$q_b^i(t + \Delta t) \approx \dot{q}_b^i(t) + q_b^i(t)\ t$$
$$= q_b^i(t)\left[1 + \frac{\underline{\omega}_{ib}^b}{z}(t)\Delta t\right] \qquad (7\text{-}11)$$

While the matrix by matrix multiplication in Eq. (7-8a) is well defined, the corresponding quaternion by quaternion multiplication indicated in Eq. (7-11) has not yet been defined.

Let us generalize the expression in the square brackets in Eq. (7-11), $[I + \underline{\omega}_{ib}^b \Delta t]$, and call it the quaternion p, where:

$$p \triangleq p_0 + \underline{p} \qquad (7\text{-}12)$$

and

$$\underline{p} \triangleq p_1\hat{i} + p_2\hat{j} + p_3\hat{k}$$

and $\hat{i}$, $\hat{j}$, $\hat{k}$ are the frame vectors (unit magnitude, mutually orthogonal, right handed) (in the particular case $\hat{i}$, $\hat{j}$, $\hat{k}$ are along the x, y, z axes of the body frame).

A good concise summary of the elements of quaternion algebra is presented in Section 2.2.1 of Reference 3. The product of two quaternions, p and q, may be written as

$$qp = q_0p_0 - \underline{q}\cdot\underline{p} + q_0\underline{p} + \underline{q}p_0 + \underline{q}x\underline{p} \tag{7-13a}$$

The preceding vector-scalar form may be handy analytically but the most efficient form for mechanization in a digital computer is as the product of a 4 x 4 matrix by a 4 x 1 vector, i.e.,

$$qp = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_o \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{7-13b}$$

The purist may object that Eq. (7-13b) is an expression for the elements of the quaternion product rather than the product per se which, of course, is correct. To be rigorous then the left hand side of Eq. (2-16) should be expressed as

$$\left[(qp)_0 \ (qp)_1 \ (qp)_2 \ (qp)_3\right]^T \ .$$

The rearranged product pq is given below:

$$pq = p_0q_0 - \underline{p}\cdot\underline{q} + p_0q + \underline{p}q_0 + \underline{p}x\underline{q} \tag{7-14a}$$

where it is seen that the sole difference is in the sign of the vector (cross) product term since $\underline{p}x\underline{q} = -\underline{q}x\underline{p}$

$$\begin{bmatrix} (pq)_0 \\ (pq)_1 \\ (pq)_2 \\ (pq)_3 \end{bmatrix} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_o \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{7-14b}$$

Equation (7-13) is perfectly general. When applied to the first order case of Eq. (7-11), we have

$$\begin{bmatrix} q_b^i(t+\Delta t)_0 \\ q_b^i(t+\Delta t)_1 \\ q_b^i(t+\Delta t)_2 \\ q_b^i(t+\Delta t)_3 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{\Delta\theta_1}{2} & -\frac{\Delta\theta_2}{2} & -\frac{\Delta\theta_3}{2} \\ \frac{\Delta\theta_1}{2} & 1 & \frac{\Delta\theta_3}{2} & -\frac{\Delta\theta_2}{2} \\ \frac{\Delta\theta_2}{2} & -\frac{\Delta\theta_3}{2} & 1 & \frac{\Delta\theta_1}{2} \\ \frac{\Delta\theta_3}{2} & \frac{\Delta\theta_2}{2} & -\frac{\Delta\theta_1}{2} & 1 \end{bmatrix} \begin{bmatrix} q_b^i(t)_0 \\ q_b^i(t)_1 \\ q_b^i(t)_2 \\ q_b^i(t)_3 \end{bmatrix} \tag{7-15}$$

Note that p is not a unit quaternion unless ω_{ib}^b is zero; hence the quaternion product, pq, will need to be normalized by dividing each element of the product by the square root of the norm, N(pq), where

$$N(pq) = (pq)_0^2 + (pq)_1^2 + (pq)_2^2 + (pq)_3^2 \tag{7-16}$$

The first order d.c.m. and quaternion updates and (ortho) normalization have been presented by way of an introduction and to point up the similarities and differences in the simplest cases. The first, second, and third order updates for each are presented in the following two subsections.

7.3 Direction Cosine Matrix Updating

The equations for higher order updating-based on the availability of incremental angles at discrete times are derived in (3) and the results are presented here.

The matrix to be updated from t to t + Δt is denoted by C(t), while the update matrix, of order i, where i = 1, 2, 3, is $M_i(t, \Delta t)$, while the resultant updated matrix is $C_i(t + \Delta t)$ or

$$C_i(t + \Delta t) = C(t)M_i(t, \Delta t) \tag{7-17}$$

The $\Delta\theta_i$ of the previous sections are shortened to θ_i, while the $\Delta\theta_i$ from the previous pass (Δt earlier) are shown as θ_i^*. Further,

$$\theta^2 \triangleq \theta_1^2 + \theta_2^2 + \theta_3^2 \tag{7-18}$$

The first order d.c.m. update (as before) is

$$C_1(t + \Delta t) = C(t)M_1(t, \Delta t) \tag{7-19}$$

where

$$M_1(t, \Delta t) = I + \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}$$

The second order d.c.m. update is

$$C_2(t + \Delta t) = C(t)M_2(t, \Delta t) \tag{7-20}$$

where

$$M_2(t, \Delta t) = M_1(t, \Delta t) + \frac{1}{2}\begin{bmatrix} -(\theta_2^2 + \theta_3^2) & \theta_1\theta_2 & \theta_1\theta_3 \\ \theta_1\theta_2 & -(\theta_3^2 + \theta_1^2) & \theta_2\theta_3 \\ \theta_1\theta_3 & \theta_2\theta_3 & -(\theta_1^2 + \theta_2^2) \end{bmatrix}$$

Note that the second order update approaches R (see Appendix C) as $\frac{\sin\theta}{\theta} \rightarrow 1$ and $\frac{1 - \cos\theta}{\theta^2} \rightarrow \frac{1}{2}$.

The third order d.c.m. update is

$$C_3(t + \Delta t) = C(t)M_3(t, \Delta t) \qquad (7\text{-}21)$$

where $M_3(t, \Delta t) = M_2(t, \Delta t)$

$$+ \frac{1}{12}\left[\begin{array}{c|c|c} 0 & 2\theta^2\theta_3 + \theta_1\theta_2^* - \theta_2\theta_1^* & -(2\theta^2\theta_2 + \theta_3\theta_1^* - \theta_1\theta_3^* \\ -(2\theta^2\theta_3 + \theta_1\theta_2^* - \theta_2\theta_1^* & 0 & 2\theta^2\theta_1 + \theta_2\theta_3^* - \theta_3\theta_2^* \\ 2\theta^2\theta_2 + \theta_3\theta_1^* - \theta_1\theta_3^* & -(2\theta^2\theta_1 + \theta_2\theta_3^* - \theta_3\theta_2^*) & 0 \end{array}\right]$$

In the third order update, the terms $(\theta_i\theta_j^* - \theta_j\theta_i^*)$ correspond to $(\underline{\omega} \times \dot{\underline{\omega}})$. It was found (5) empirically that the inclusion of these terms in regimes of very high angular acceleration, $\dot{\omega}$, degraded the algorithm performance. Accordingly these terms may be omitted at the user's discretion. (they are presently "commented" out of the third order d.c.m. update.)

The program flow for the d.c.m. update utilizes Eqs. (7-19), (7-20), and (7-21) in order, as required, followed by Eq. (7-17), and orthonormalization using Eq. (7-10).

7.4 Quaternion Updating and DCM Generation

Regardless of the order of the update, the form follows Eq. (7-13), while the equations are derived in (5). The quaternion to be updated from t to t + Δt is denoted by $q(t)$, while the updating quaternion, of order i, where i = 1, 2, 3, is $p^{(i)}(t, \Delta t)$ and the updated quaternion is $q^{(i)}(t, \Delta t)$ or

$$q^{(i)}(t + \Delta t) = q(t)p^{(i)}(t, \Delta t) \tag{7-22}$$

The elements of $p^{(i)}(t, \Delta t)$ are all functions of the $\Delta\theta_i/2$, shortened to $\theta_i/2$, while the $\Delta\theta_i/2$ from the previous pass are shown as $\theta_i^*/2$. Similarly, θ^2 from the d.c.m. case becomes

$$\left(\frac{\theta}{2}\right)^2 = \frac{\theta}{2}\cdot\frac{\theta}{2} = \left(\frac{\theta_1}{2}\right)^2 + \left(\frac{\theta_2}{2}\right)^2 + \left(\frac{\theta_3}{2}\right)^2 \tag{7-23}$$

The first order update quaternion (as before) is

$$p^{(1)}(t, \Delta t) = 1 + \frac{\theta}{2} \tag{7-24}$$

or

$$p_0^{(1)} = 1$$

$$p_1^{(1)} = \frac{\theta_1}{2}$$

$$p_2^{(1)} = \frac{\theta_2}{2}$$

$$p_2^{(1)} = \frac{\theta_2}{2}$$

The second order update quaternion is

$$p^{(2)}(t, \Delta t) = p^{(1)}(t, \Delta t) - \frac{1}{2}\left(\frac{\theta}{2}\cdot\frac{\theta}{2}\right)$$

$$= p^{(1)}(t, \Delta t) - \frac{1}{2}\left(\frac{\theta}{2}\right) \tag{7-25a}$$

Since the second order portion of the update is a scalar it affects only p_0, so

$$p_0^{(2)} = p_0^{(1)} - \frac{1}{2}\left(\frac{\theta}{2}\right)^2 = 1 - \frac{1}{2}\left(\frac{\theta}{2}\right)^2$$

$$p_i^{(2)} = p_i^{(1)}, \; i = 1, 2, 3 \qquad \text{(7-25b)}$$

The third order update quaternion is

$$p^{(3)}(t, \Delta t) = p^{(2)}(t, \Delta t) - \frac{1}{6}\left[\left(\frac{\theta}{2} \cdot \frac{\theta}{2}\right)\frac{\theta}{2} + \left(\frac{\theta}{2} \times \frac{\theta}{2}\right)\right] \qquad \text{(7-26a)}$$

Since the third order update is a vector, it does not affect p_o, so

$$p_o^{(3)} = p_0^{(2)}$$

$$p_1^{(3)} = p_1^{(2)} - \frac{1}{6}\left[\left(\frac{\theta}{2}\right)^2 \frac{\theta_1}{2} + \left(\frac{\theta_2}{2}\frac{\overset{*}{\theta}_3}{2} - \frac{\theta_3}{2}\frac{\overset{*}{\theta}_2}{2}\right)\right]$$

$$p_2^{(3)} = p_2^{(2)} - \frac{1}{6}\left[\left(\frac{\theta}{2}\right)^2 \frac{\theta_2}{2} + \left(\frac{\theta_3}{2}\frac{\overset{*}{\theta}_1}{2} - \frac{\theta_1}{2}\frac{\overset{*}{\theta}_3}{2}\right)\right]$$

$$p_3^{(3)} = p_3^{(3)} - \frac{1}{6}\left[\left(\frac{\theta}{2}\right)^2 \frac{\theta_3}{2} + \left(\frac{\theta_1}{2}\frac{\overset{*}{\theta}_2}{2} - \frac{\theta_2}{2}\frac{\overset{*}{\theta}_1}{2}\right)\right] \qquad \text{(7-26b)}$$

The similarity between the third order terms of the update for the d.c.m. and quaternion should be noted. The main difference is the use of $\theta_i/2$ in the quaternion, whereas θ_i appears in the d.c.m.

Obviously, the simplest way to formulate the algorithm to implement Eq. (7-13b) would be to form the four elements of the update quaternion, $p^{(i)}$, of order i - i.e., start with first order, add second and third order terms as required, form the 4 x 4 matrix from the elements of $p^{(i)}$, and perform the indicated matrix by vector [(4 x 4) x (4 x 1)] multiplication.

The V-A module, uses three separate inline-coded, matrix by vector multiplication routines - one for each order of quaternion update. However, the algorithm used is correct, even though it takes about twice as many statements as the simpler and more elegant algorithm (see reference (5), in subroutine HSINTG).

After forming the updated quaternion, $p^{(i)}(t, \Delta t)$, it is periodically normalized by dividing each element by the square root of the norm as indicated in Eq. (7-16) that is

$$d \triangleq \left(q_0^2 + q_1^2 + q_2^2 + q_3^2\right)^{1/2}$$

$$qi_{norm} = q_i/d, \; i = 0, 1, 2, 3 \qquad (7\text{-}27)$$

where the q's are elements of the updated but unnormalized quaternion, and $q_{i_{norm}}$ are the elements of the updated, normalized quaternion.

Next, the quaternion is converted to a d.c.m., as shown in Appendix C, to obtain $C_{b_{norm}}^{i\,(i)}(t + \Delta t)$. Omitting the subscripts, superscripts, argument and comments, the d.c.m. is

$$C_{b_{norm}}^{i\,(i)}(t + \Delta t)^{*} = \left[\begin{array}{c|c|c} 1 - 2\left(q_2^2 + q_3^2\right) & 2\left(q_1q_2 - q_3q_0\right. & 2(q_3q_1 + q_2q_0\left.\right) \\ 2\left(q_3q_0 + q_1q_2\right) & 1 - 2\left(q_3^2 + q_1^2\right) & 2\left(q_2q_3 - q_0q_1\right) \\ 2\left(q_3q_1 - q_2q_0\right) & 2\left(q_2q_3 + q_0q_1\right) & 1 - 2\left(q_1^2 + q_2^2\right) \end{array}\right]$$

(7-28)

This completes the portions of the V-A module which depend on the particular kind of update.

*In subsequent sections, it is more convenient to use "t" as the present module operating time, rather than (t + Δt).

7.5 Incremental Velocity Transformation

Once C_b^i has been obtained by either of the foregoing means, it is employed to transform the incremental velocities - accumulated from $(t - \Delta t)$ to t - from the body frame to the inertial frame. The equation which is actually mechanized is

$$\underline{\Delta V}^i(t) = \hat{C}_b^i(t - \frac{\Delta t}{2})\underline{\Delta V}^b(t) \tag{7-29}$$

where $\underline{\Delta V}^b(t)$ is given by

$$\underline{\Delta V}^b(t) \triangleq \int_{t - \Delta t}^{t} \underline{a}^b(\tau)d\tau \tag{7-30}$$

The approximate mid-computation cycle value of the body-to-inertial transformation, $\hat{C}_b^i(t - \frac{\Delta t}{2})$, is obtained simply by averaging the updated C_b^i with the "old" value from the previous computation cycle i.e.

$$^*\ \hat{C}_b^i(t - \frac{\Delta t}{2}) \triangleq \frac{1}{2}\left[C_b^i(t) + C_b^i(t - \Delta t)\right] \tag{7-31}$$

While the approximation in (7-31) is only first order (in $\Delta\theta$), its effects are not cumulative.

The final operation performed on the $\underline{\Delta V}$'s before output is to transform them from the i-frame to the i' frame (the inertial frame used by the navigation algorithm), and to sum them, if the navigation algorithm is operating on a longer computation cycle. The transformation of $\underline{\Delta V}$'s is

$$\underline{\Delta V}^{i'}(t) = C_i^{i'}\underline{\Delta V}^i(t) \tag{7-32}$$

*This transformation is denoted by DCMMID in program mnemonics.

where

$$C_i^{i'} = C_c^{c'} = \tilde{Z}(\pi/2)\tilde{X}(\pi/2) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^{*}$$

or

$$\begin{bmatrix} \Delta V_1^{i'} \\ \Delta V_2^{i'} \\ \Delta V_3^{i'} \end{bmatrix} = \begin{bmatrix} \Delta V_3^{i} \\ \Delta V_1^{i} \\ \Delta V_2^{i} \end{bmatrix} \qquad (7\text{-}32b)$$

7.6 Body to "New" Earth-Fixed Frame Transformation, $C_b^{e'}$

For attitude computation the updated C_b^i must be transformed to the "new" earth fixed or e'-frame, used in the navigation algorithm. It was shown, in Appendix B, that this added transformation, $C_i^{e'}$ is given by

$$C_i^{e'} = Z(\omega_{ie}t)\tilde{Z}(\pi/2)\tilde{X}(\pi/2)$$

$$= \begin{bmatrix} s\omega_{ie}t & 0 & c\omega_{ie}t \\ c\omega_{ie}t & 0 & -s\omega_{ie}t \\ 0 & 1 & 0 \end{bmatrix} \qquad (7\text{-}33)$$

where the value of t used in Eq. (7-33) corresponds to $(t + \Delta t)$ in the notation of this section.

The transformation which is output to the navigation module, $C_b^{e'}$, or DCM in the V-A mnemonics is

* See Appendix B.

$$C_b^{e'} = C_i^{e'} C_b^{i'} \qquad (7\text{-}34)$$

The sum of the $\underline{\Delta V}^i$'s since the last navigation computation cycle are also output to the navigation module.

SECTION 8

8. LOCAL LEVEL NAVIGATION ALGORITHM AND ALTITUDE COMPUTATION FOR A STRAPDOWN INERTIAL NAVIGATOR

The navigation/attitude algorithm is the third or final module of the three main modules which comprise the navigation (and attitude) software for a SD INS.

The function of this module is to compute the velocity, position, and attitude (roll, pitch and heading) of the vehicle at the end of each "slow" computation cycle -- on the order of 2 to 20 Hz.

To accomplish this, the module is initialized at the start of navigation ($t = 0$) on the basis of velocity and position data supplied by the trajectory module.

During each succeeding (non-initialization) pass, the module accepts incremental velocities, in an inertial reference frame, summed over the slow cycle, from the velocity and attitude algorithm, and the barometric altitude from the altimeter module; this is all the information required for velocity and position computation.

For attitude computation, the module accepts the direction cosine matrix relating the body frame to the "new" earth-fixed frame from the velocity and attitude module and combines it with the DCM relating the "new" earth-fixed frame to the computational frame (the "position" matrix) and extracts the computed roll, pitch and yaw. The latter is corrected by the wander angle (azimuth of the computational frame) to obtain the heading of the vehicle.

8.1 Analytical Development of the Local Vertical Wander Azimuth Navigation and Attitude Algorithms

The actual form in which the equations for the computation of the earth-referenced velocity, position and attitude of a terrestrial inertial navigation are mechanized in an airborne digital computer has been the object of considerable study for the past quarter of a century.

A survey of the algorithms employed in several aircraft INS's and the algorithms recommended for future use (in tactical aircraft) was covered in some detail in (R-977). The model for the earth and its gravitational field is that of the WG2-72 ellipsoidal earth (R-977); many of the equations relevant to this model are presented in PROFGEN, which is the mission profile generator that ultimately will be used to drive the simulator.

In terms of the first reference above, the navigation and attitude computations used in the INSS most nearly approximate those that would be employed with a space stable inertial measurement unit (SS IMU), i.e., $\underline{\Delta V}$'s are accumulated in an inertial frame, with a local vertical wander azimuth (LVWA) computational frame, i.e., the equations of motion are actually integrated in the LVWA frame. Still in terms of the first reference, the algorithms used in the INSS correspond to the upgraded algorithm. It should be noted that the sign of the wander angle used in the navigation algorithm has been reversed with respect to the reference to conform with the remainder of the program.

With these preliminaries, the equations mechanized in the navigation and attitude computations are presented. Note that the first section is extracted directly from (R-977).

8.2 Navigation Equations in Local Vertical Frame

The fundamental equation of inertial navigation for a navigator employing an earth relative computational frame (c-space) is given by

$$\dot{\underline{v}}^c = \underline{f}^c + \underline{g}^c - (\Omega^c_{ec} + 2\Omega^c_{ie})\ \underline{v}^c \qquad (8-0)$$

where $\underline{v}^c$ is the earth relative velocity in computational frame

$\dot{\underline{v}}^c$ is the time rate of change of $\underline{v}^c$

$\underline{f}^c$	is the specific force (the portion of inertial acceleration sensed by the accelerometers) expressed in computational frame coordinates
$\underline{g}^c$	$\triangleq \underline{G}^c - \Omega^c_{ie}\,\Omega^c_{ie}\,\underline{r}^c$, is the resultant of mass attraction, $\underline{G}^c$, and centripetal force, $-\Omega^c_{ie}\,\Omega^c_{ie}\,\underline{r}^c$, expressed in computational frame coordiantes
e	is the earth-fixed frame which is rotating at a constant angular rate, ω_{ie}, with respect to the inertial frame
i	is the earth-centered inertial frame which is regarded as non rotating with respect to the "fixed" stars
Ω^c_{ec}	is the skew-symmetric form of the angular velocity of the computational frame with respect to the earth-fixed frame, expressed in computational frame coordinates, $\underline{\omega}^c_{ec}$.
Ω^c_{ie}	is the skew-symmetric form of the angular velocity of the earth-fixed frame with respect to the inertial frame, expressed in computational frame coordiantes, $\underline{\omega}^c_{ie}$. The magnitude of $\underline{\omega}^c_{ie}$, ω_{ie}, is a constant but the components vary as a function of latitude (and wander angle).

The function of the navigation computer may be considered to be the integration of equation (8-0), to obtain the earth relative velocity, $\underline{v}^c$, followed by a second integration to obtain the earth relative position. Since the earth relative position is expressed mainly in angular coordinates, latitude, longitude, (and wander angle), it is necessary to convert the level components of linear velocity from the first integration into angular velocity form by dividing by the appropriate variable, radii of curvature to obtain $\underline{\omega}^c_{ec}$ which is then converted to the skew-symmetric form, Ω^c_{ec}. The differential equation which is integrated to obtain the angular position is $\dot{C}^3_c = C^e_c\,\Omega^c_{ec}$ (8-0a)

The actual latitude, longitude (and wander angle) are extracted from C_c^e via the appropriate inverse trigonometric functions. No primes (') have been added to the c, e, i frames, since usually one each computational, earth-fixed, and inertial frame is sufficient. However, the primes are employed in the following presentation to differentiate the "new" c, e, and i frames from the "old" c, e, and i frames.

The remaining position variable, altitude is obtained by integrating the vertical component of $\underline{v}^c$. An external altitude reference is required to eliminate the inherent instability of any <u>pure</u> inertial vertical channel. Thus, the minimum number of scalar integrations required to mechanize a local vertical wander azimuth navigator is <u>thirteen</u> (13). However, only five of the nine elements of C_c^e are employed directly in the navigation computations so frequently only six of the nine elements are computed.

8.3 Navigation Initialization

On the initialization pass, at t = 0, the navigation module obtains the initial position and velocity from the trajectory module, while the initial position errors are supplied from the module's own initialization file.

The initial value of the (LVWA) computational frame, c', to "new" earth-fixed frame, e', transformation, $C_{c'}^{e'}$, or A, is computed from

$$C_{c'}^{e'} = A = \tilde{Z}(\ell)\ Y(L)\ \tilde{X}(\alpha)$$

$$= \left[\begin{array}{c|c|c} cLc\ell & -c\alpha s\ell - s\alpha sLc\ell & s\alpha s\ell - c\alpha sLc\ell \\ cLs\ell & c\alpha c\ell - s\alpha sLs\ell & -s\alpha c\ell - c\alpha sLs\ell \\ sL & s\alpha cL & c\alpha cL \end{array}\right] \tag{8-1}$$

where	L	is the initial latitude plus errors
	ℓ	is the initial longitude plus errors
	α	is the initial wander azimuth

This matrix is the repository of the (horizontal) position information for the navigation algorithm. It is developed in Appendix B.

8.4 Initial Velocity

The initial earth-relative velocities in an ENU coordinate frame, $\underline{v}^{\ell}$, are computed with their initial errors and transformed to the LVWA computational frame, c'.

$$\underline{v}^{\ell}_{nav} = \underline{v}^{\ell} + \underline{\varepsilon v}^{\ell}$$

$$\underline{v}^{c'}_{nav} = C^{c'}_{\ell} \, \underline{v}^{\ell}_{nav} \qquad (8\text{-}2)$$

where $C^{c'}_{\ell} = C^{c'}_{\ell'} \, C^{\ell'}_{\ell}$

and $C^{c'}_{\ell'} = X(\alpha)$

and $C^{\ell'}_{\ell} = \tilde{Z}(\pi/2) \, \tilde{X}(\pi/2)$

(See Appendix B).

The form in which equation 8-2 is mechanized is

$$\underline{v}^{c'}_{nav} = X(\alpha)\left[\tilde{Z}(\pi/2) \, \tilde{X}(\pi/2) \, \underline{v}^{\ell}_{nav}\right] \qquad (8\text{-}2a)$$

since the quantity if square brackets merely respresents a rearrangement of the elements of $\underline{v}^{\ell}_{nav}$.

8.5 Initial Angular Rate, $\underline{\omega}^{c'}_{e'c'}$

The initial radii of curvature are computed using the exact formulae and the parameters of the WGS72 ellipsoid

$$r_p = h_{nav} + \frac{r_e}{\left(1-\varepsilon^2 A_{31}^2\right)^{1/2}} \qquad (8\text{-}3)$$

and $$r_m = h_{nav} + \frac{r_e\left(1-\varepsilon^2\right)}{\left(1-\varepsilon^2 A_{31}^2\right)^{3/2}} \tag{8-4}$$

where r_e	is the earth's equatorial radius
ε^2	is the square of the eccentricity of the meridional ellipse
$A_{31}=sL$	is the sine of latitude
$h_{nav}=h(0)+\varepsilon h$	is the loaded value of the altitude above sea level
r_p	is the radius of curvature of the prime vertical ellipse
r_m	is the radius of curvature of the meridional ellipse

The angular velocities about east and north, ω_E, ω_N, are computed from

$$\omega_E = \frac{-v_N}{r_m} \tag{8-5}$$

and $$\omega_N = \frac{v_E}{r_p} \tag{8-6}$$

where in general,

$$v_E = v_2^{c'}\ C\alpha - v_3^{c'}\ S\alpha$$

and $$v_N = v_2^{c'}\ S\alpha + v_3^{c'}\ C\alpha$$

The angular velocities about the horizontal axes of the computational frame are computed by transforming the ω_E and ω_N from the ℓ' frame to the c' frame

$$\underline{\omega}^{c}_{e'c'} \overset{\Delta}{=} \underline{\rho} = X(\alpha) \begin{bmatrix} 0 \\ \omega_E \\ \omega_N \end{bmatrix}, \quad \omega_U, \overset{\Delta}{=} 0 \tag{8-7}$$

where the ρ_i are the components of the angular velocity of the computational frame in the new "earth-fixed frame".

Equations 8-3 through 8-7 are mechanized in a subroutine (ANGVEL).

The components of the earth's sidereal rate, $\underline{\omega}^{c'}_{i'e'}$, in the computational frame are

$$\underline{\omega}^{c'}_{i'e'} = C^{c'}_{i'} \begin{bmatrix} 0 \\ 0 \\ \omega_{ie} \end{bmatrix}$$

$$= C^{c'}_{i'}\, {}^{e}_{3}\omega_{ie}$$

$$= C^{c'}_{e'} e_3 \omega_{ie}$$

$$= \begin{bmatrix} \tilde{e}_3 C^{e'}_{c'} \end{bmatrix}^T \omega_{ie}$$

$$= \begin{bmatrix} \tilde{e}_3 A \end{bmatrix} \omega_{ie} \qquad (8\text{-}8)$$

where e_3 is a unit vector whose elements are $[0, 0, 1]^T$.

All equation 8-8 says is the elements of $\underline{\omega}^{c'}_{i'e'}$, are formed from the elements of the third row of the A-matrix and the earth's sidereal rate, ω_{ie}, which latter is directed along the Z axis of the i' and e'-frames.

The total angular velocity of the "new" computational ('c) frame with respect to the "new" inertial (i') frame, in the "new" computational frame, $\underline{\omega}^{c'}_{i'c'}$, is

$$\underline{\omega}^{c'}_{i'c'} = \underline{\omega}^{c'}_{i'e'} + \underline{\omega}^{c}_{i'c'} \qquad (8\text{-}9)$$

and are the gyro torquing rates for a gimballed LVWA IMU. In the present SD case, these become the "matrix" torquing signals to update the $C_{e'}^{c'}$ for the $\underline{\Delta v}$ tranformation.

Coriolis and centripetal compensation requires, $\underline{\omega}_{cor}$, where

$$\begin{aligned} \underline{\omega}_{cor} &= \underline{\omega}_{e'c'}^{c'} + 2\underline{\omega}_{i'e'}^{c'} \\ &= \underline{\omega}_{i'c'}^{c'} + \underline{\omega}_{i'e'}^{c'} \end{aligned} \qquad (8\text{-}10)$$

and the form of the correction is $\underline{\omega}_{cor} \times \underline{v}^{c'}$ or (see equation (8-0)

$$(\underline{\omega} \times \underline{v}) = \underline{\omega}_{cor} \times \underline{v}^{c'} = \Omega_{cor}\, \underline{v}^{c'} \qquad (8\text{-}11)$$

$$= \left[\begin{array}{c|c|c} 0 & -\omega_{cor_3} & \omega_{cor_2} \\ \omega_{cor_3} & 0 & -\omega_{cor_1} \\ -\omega_{cor_2} & \omega_{cor_1} & 0 \end{array}\right] \begin{bmatrix} v_1^{c'} \\ v_2^{c'} \\ v_3^{c'} \end{bmatrix}$$

Since the correction is applied at the velocity level all the elements are multiplied by the computation cycle, Δt.

Equations 8-8 through 8-11 are mechanized in a subroutine (TORCOR).

Several other computations are required in preparation for the first normal pass through the navigation algorithm.

First, the computational frame to earth fixed transformation, computed at t=0, A(0), extrapolated to the middle of the first computational cycle, using the initial values of the computational frame. $\ell(0)$; or $\underline{\omega}^{c'}_{e'c'}(0)$, and a second under update.

$$\hat{A}\left(\frac{\Delta t}{2}\right) = A(0) \left\{I + \Omega^{c'}_{e'c'}(0) \left(\frac{\Delta t}{2}\right) + \left[\Omega^{c'}_{e'c'}(0)\right]^2 \frac{1}{2} \left(\frac{\Delta t}{2}\right)^2\right\}$$

where

$$\Omega^{c'}_{e'c'}(0) = \begin{bmatrix} 0 & -\rho_3(0) & \rho_2(0) \\ \rho_3(0) & 0 & 0 \\ -\rho_2(0) & 0 & 0 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 0 \\ \rho_2(0) \\ \rho_3(0) \end{bmatrix} \quad \underline{\Delta} \quad \underline{\omega}^{c'}_{e'c'}(0) \tag{8-12}$$

and Δt is the navigation computation cycle.

The expression in the curly braces is computed in subroutine AUP, which accepts as inputs θ_y and θ_z where (in this case)

$$\theta_y = \rho_2(0)\frac{\Delta t}{2}$$

$$\theta_z = \rho_3(0)\frac{\Delta t}{2}$$

and the update matrix

$$I + \Theta + \frac{1}{2}\Theta^2 = \left[\begin{array}{c|c|c} 1 - \left(\frac{\theta_y^{\,2} + \theta_z^{\,2}}{2}\right) & -\theta_z & \theta_y \\ \theta_z & 1-\frac{\theta_z^{\,2}}{2} & \theta_y\theta_z \\ -\theta_y & \theta_y\theta_z & 1-\frac{\theta_y^{\,2}}{2} \end{array}\right] \tag{8-13}$$

Based on the initial values of latitude east velocity and angle, the latter is extrapolated to the middle of the first computation cycle using a first order algorithm.

$$\sin\left[\hat{\alpha}\left(\frac{\Delta t}{2}\right)\right] \doteq \sin\left[\alpha(0)\right] + \delta\left[\alpha\left(\frac{\Delta t}{2}\right)\right]\cos\left[\alpha(0)\right]$$

$$\cos\left[\hat{\alpha}\left(\frac{\Delta t}{2}\right)\right] \doteq \cos\left[\alpha(0)\right] - \delta\left[\alpha\left(\frac{\Delta t}{2}\right)\right]\sin\left[\alpha(0)\right] \qquad (8\text{-}14)$$

$$\text{where } \delta\left[\alpha\left(\frac{\Delta t}{2}\right)\right] \underline{\underline{\Delta}} \ \alpha\left(\frac{\Delta t}{2}\right) - \alpha(0) \doteq \dot{\alpha}(0)\frac{\Delta t}{2}$$

$$\doteq \frac{V_E(0)\sin L(0)}{r_E \cos L(0)}\frac{\Delta t}{2}$$

Mid-computation cycle altitude is obtained by extrapolation of the initial value and the initial vertical velocity

$$\hat{h}\left(\frac{\Delta t}{2}\right) = h(0)\ V_u(0)\ \frac{\Delta t}{2} \qquad (8\text{-}15)$$

Both the extrapolated A-matrix, $\hat{A}\left(\frac{\Delta t}{2}\right)$ and the extrapolated altitude, $\hat{h}\left(\frac{\Delta t}{2}\right)$ are employed in the computation of gravity at $\frac{\Delta t}{2}$, in the subroutine, GRAV. Two components exist - the north-south and the vertical components. The east-west component is zero. However, the level axes of the computational frame are not generally north-south and east-west and must be resolved through the wander angle (or azimuth of the computational frame).

The vertical (up) component of gravity is

$$\hat{g}_1\left(\frac{\Delta t}{2}\right) = -\left[g_o + g_{L_1}\sin^2\hat{L}\left(\frac{\Delta t}{2}\right) + g_{L_2}\sin^4\hat{L}\left(\frac{\Delta t}{2}\right)\right]$$

$$\times\left\{1 - \left[g_{h_1} + g_{h_2}\sin^2\hat{L}\left(\frac{\Delta t}{2}\right)\right]\hat{h}\left(\frac{\Delta t}{2}\right) + g_{h_3}\hat{h}^2\left(\frac{\Delta t}{2}\right)\right\} \qquad (8\text{-}16a)$$

$$\text{where} \quad \sin^2\hat{L}\left(\frac{\Delta t}{2}\right) = \hat{A}^2_{31}\left(\frac{\Delta t}{2}\right)$$

The level component, g_2 and g_3, which are east and north for $\alpha=0$ are

$$\hat{g}_2\left(\frac{\Delta t}{2}\right) = g_h \hat{h}\left(\frac{\Delta t}{2}\right) \sin \hat{L}\left(\frac{\Delta t}{2}\right) \cos \hat{L}\left(\frac{\Delta t}{2}\right) \sin \hat{\alpha}\left(\frac{\Delta t}{2}\right) \qquad (8\text{-}16b)$$

$$\hat{g}_3\left(\frac{\Delta t}{2}\right) = g_n \hat{h}\left(\frac{\Delta t}{2}\right) \sin \hat{L}\left(\frac{\Delta t}{2}\right) \cos \hat{L}\left(\frac{\Delta t}{2}\right) \cos \hat{\alpha}\left(\frac{\Delta t}{2}\right) \qquad (8\text{-}16c$$

where $\cos \hat{L}\left(\frac{\Delta t}{2}\right) \sin \hat{\alpha}\left(\frac{\Delta t}{2}\right) = \hat{A}_{32}\left(\frac{\Delta t}{2}\right)$

$\cos \hat{L}\left(\frac{\Delta t}{2}\right) \cos \hat{\alpha}\left(\frac{\Delta t}{2}\right) = \hat{A}_{32}\left(\frac{\Delta t}{2}\right)$

and $g_o = 32.0877057 \text{ ft/sec}^2$

$g_{L_1} = 0.16939081 \text{ ft/sec}^2$

$g_{L_2} = 7.5281\text{x}10^{-4} \text{ ft/sec}^2$

$g_{h_1} = 9.6227\text{x}10^{-8} \text{ ft}^{-1}$

$g_{h_2} = 6.4089\text{x}10^{-10} \text{ ft}^{-1}$

$g_{h_3} = 6.8512\text{x}10^{-15} \text{ ft}^{-2}$

$g_n = 1.63\text{x}10^{-8} \text{ ft}^2/\text{sec}^2$

The extrapolated A matrix is also used to recompute the Corolis corrections (call TORCOR) more nearly appropriate to the mid-computation cycle.

Finally, the incremental velocities in the "new" inertial frame, $\underline{\Delta v}^i(0)$ are set to zero, i.e.,

$$\underline{\Delta v}^i(0) = 0$$

These are DVI in program mnemonics. The initialization of the navigation algorithm is complete.

8.6 LVWA Navigation and Attitude Computations for Normal Computation Cycle

The following operations are performed every navigation and attitude) computation cycle following initialization. While Δt is used for this module as well as all the others, it does not necessarily follow that all the Δt's are the same. What is necessary is: the Δt for "later" modules must be the same as or an integral multiple of, the computation cycle of all the "earlier" moudles which supply the input data to it.

Note that in the navigation algorithm the inertial, earth-fixed, and computational frames are all the "new" or primed frames, unless stated otherwise.

1) Transform Δv's from the Inertial to the Computational Frame

The effect of the two-step process is to mechanize

$$\underline{\Delta v}^{c'}(t) = C_{i'}^{c'}(t-\frac{\Delta t}{2})\,\underline{\Delta v}^{i'}(t)$$

$$= C_{e'}^{c'}(t-\frac{\Delta t}{2})\left[C_{i'}^{e'}(t-\frac{\Delta t}{2})\,\Delta v^{i'}(t)\right]$$

or $$\Delta v^{e'}(t) = C_{i'}^{e'}(t-\frac{\Delta t}{2})\,\Delta v^{i'}(t) \qquad (8\text{-}17)$$

where $C_{L'}^{e'}(t-\frac{\Delta t}{2}) = Z\left[\omega_{i_e}(t-\frac{\Delta t}{2})\right]$

and $\Delta v^{c'}(t) = C_{e'}^{c'}(t-\frac{\Delta t}{2})\,\underline{\Delta v}^{e'}(t)$

where $C_{e'}^{c'}(t-\frac{\Delta t}{2}) = \tilde{A}(t-\frac{\Delta t}{2})$

and $\tilde{A}(t-\frac{\Delta t}{2})$ is the transpose of the extrapolated value of the "new" computational to "new" earth-fixed frame transformation generated during the preceding computation cycle (or in initialization). Strictly speaking it should be denoted as $\tilde{\hat{A}}$.

Compute Components of Gravity in the LVWA Computational Frame

The components of gravitational and centripetal force at (t-Δt) are computed using the WGS72 ellipsoidal formula, and the values of L, α, h.

$$g_1^{c'}(t-\Delta t) = -\left[C_{g_0} + C_{g_2} A_{31}^2(t-\Delta t) + C_{g_4} A_{31}^4(t-\Delta t)\right]$$

$$+\left\{1-\left[C_{h_1} - C_{h_4} A_{31}^2(t-\Delta t)\right] h(t-\Delta t) + C_{h_2} h^2(t-\Delta t)\right\}$$

$$g_2^{c'}(t-\Delta t) = C_{h_0} A_{31}(t-\Delta t) A_{32}(t-\Delta t) h(t-\Delta t)$$

2) Update LVWA Velocity

The Δv, Coriolis, and gravity terms are used to updated LVWA velocity (per equation 8-0)

$$\underline{v}^{c'}(t) = \underline{v}^{c'}(t-\Delta t) + \Delta v^{c'}(t) + \underline{\omega}_{cor}(t-\frac{\Delta t}{2}) \times \underline{v}^{c'}(t-\Delta t)\ t + g^{c'}(t-\frac{\Delta t}{2})\Delta t \tag{8-18}$$

3) Add Vertical Damping Term (to Vertical Velocity)

$$v_1^{c'}(t) = v_1^{c'}(t) + \left[\hat{\delta}\alpha(t-\frac{\Delta t}{2}) + C_{vd2}\ \delta h(t-\frac{\Delta t}{2})\right] \Delta t \tag{8-19}$$

where $C_{vd1} = 1.62\text{x}10^{-3}\text{sec}^{-2}$

$$\hat{\delta}\alpha(t-\frac{\Delta t}{2}) = C_{vd3} \int_0^{t-\Delta t} \delta h(\tau) d\tau$$

$$\delta h(t-\frac{\Delta t}{2}) = \hat{h}_B(t-\frac{\Delta t}{2}) - \hat{h}(t-\frac{\Delta t}{2})$$

$\hat{\delta}\alpha$ and δh are extrapolated values from the previous computation cycle.

4) Interpolated and Extrapolate Computational Frame Velocity

The velocity extrapolated to the middle of the next comp. cycle is

$$\hat{\underline{v}}^{c'}(t-\frac{\Delta t}{2}) = \frac{3}{2}\underline{v}^{c'}(t) - \frac{1}{2}\underline{v}^{c'}(t-\Delta t) \qquad (8\text{-}20)$$

The interpolated velocity is

$$\hat{\underline{v}}^{c'}(t-\frac{\Delta t}{2}) = \frac{1}{2}\left[\underline{v}^{c'}(t) + \underline{v}^{c'}(t-\Delta t)\right] \qquad (8\text{-}21)$$

5) Compute Angular Velocity, f

The angular rates of the computational frame with respect to the earth-fixed frame expressed in the computational frame coordinates at the mid point of the comp. cycle, $\underline{\omega}_{e'c'}^{c'}(t-\frac{\Delta t}{2})$, are required to update the A matrix, $C_{c'}^{e'}$, from $(t-\Delta t)$ to t.

The following inputs are supplied to the angular velocity subroutine, ANGVEL:

$$\underline{v}^{c'}(t-\frac{\Delta t}{2})$$

$$h\ (t-\frac{\Delta t}{2})$$

$$\sin\alpha\ (t-\frac{\Delta t}{2})$$

$$\cos\alpha\ (t-\frac{\Delta t}{2})$$

$$\sin^2 L\ (t-\frac{\Delta t}{2})$$

All except the first are the extrapolated values from the previous cycle, while the first is the interpolated value from the present cycle.

The subroutine computes f $(t-\frac{\Delta t}{2})$ using Equations 8-3 through 8-7.

6) <u>Update Transformation from Computational to Earth Fixed Frame</u> (from t-Δt) to t).

The values of $\rho(t-\frac{\Delta t}{2})$ are converted to angular form by multiplying by Δt, i.e:

$$\rho_1(t-\frac{\Delta t}{2})\ \Delta t \triangleq 0$$

$$\rho_2(t-\frac{\Delta t}{2})\ \Delta t = \theta_y$$

$$\rho_3(t-\frac{\Delta t}{2})\ \Delta t = \theta_z$$

The second order update matrix is formed by subroutine AUP, using θ_y and θ_z above as arguments

The updated A-matrix, $C^{e'}_{c'}(t)$ is

$$A(t) \triangleq C^{e'}_{c'}(t) = C^{e'}_{c'}(t-\Delta t)\ \left[I + \Theta + \frac{1}{2}\Theta\right]^2 \qquad (8\text{-}22)$$

where the expression in square braces is defined by Equation 8-13.

7) <u>Orthonormalize $A^{(2)}(t)$</u>

The second order updated DCM, $A^{(2)}(t)$, is orthonormalized every NORTH computation cycle.

$$A_n(t) = A^{(2)}(t) - \frac{1}{2}A^{(2)}(t)\left[\tilde{A}^{(2)}(t)\ A^{(2)}(t) - I\right] \qquad (8\text{-}25)$$

The orthonormalized form, $A_n(t)$, is restored as A(t).

8) <u>Extrapolate A-matrix to Mid Computation Cycle</u> (i.e. from t to t + $\frac{\Delta t}{2}$)

In (6) the A-matrix was updated to t, A(t), using the angular velocities appropriate to $(t-\frac{\Delta t}{2})$, $\rho(t-\frac{\Delta t}{2})$. Now, A(t) is extrapolated using the same $\rho(t-\frac{\Delta t}{2})$ and the same subroutine, AUP, with arguments

$$\rho_1(t-\frac{\Delta t}{2})\frac{\Delta t}{2} \triangleq 0$$

$$\rho_2(t-\frac{\Delta t}{2})\frac{\Delta t}{2} = \theta_y$$

$$\rho_3(t-\frac{\Delta t}{2})\frac{\Delta t}{2} = \theta_z$$

The extrapolated A-matrix is

$$A(t+\frac{t}{2}) = C_{c'}^{e'}(t+\frac{t}{2}) = C_{c'}^{e'}(t)\left[I + \Theta + \frac{1}{2}\Theta\right]^2 \qquad (8\text{-}24)$$

where
$$\Theta = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & 0 \\ -\theta_y & 0 & 0 \end{bmatrix}$$

9) Compute Altitude and Perform Vertical Damping Calculations

The altitude is updated not only to account for the mean vertical velocity over the computation cycle, $\hat{\underline{v}}_1^{c'}(t-\frac{\Delta t}{2})$, but also to include a term proportional to the difference between the external and inertially - derived altitude over the same period, $\delta h(t-\frac{\Delta t}{2})$. The equation mechanized is

$$h(t) = h(t-\Delta) + \left[\hat{v}_1^{c'}(t-\frac{\Delta t}{2}) + C_{vd_1}\,\delta h(t-\frac{\Delta t}{2})\right]\Delta t \qquad (8\text{-}25)$$

The altitude is extrapolated to the midpoint of the next comp. cycle by

$$\hat{h}(t+\frac{\Delta t}{2}) = h(t) + v_1^{c'}(t)\,\frac{\Delta t}{2} \qquad (8\text{-}26)$$

Similarly, the external (barometric) altitude, h_B, is extrapolated to the same time using

$$\hat{h}_B(t+\frac{\Delta t}{2}) = \frac{3}{2}\;h_B(t) - \frac{1}{2}\;h_B(t-\Delta t) \qquad (8\text{-}27)$$

The predicted altitude error at $(t+\frac{\Delta t}{2})$, $\hat{\delta}h(t+\frac{\Delta t}{2})$ is

$$\hat{\delta}h(t+\frac{t}{2}) = \hat{h}_B(t+\frac{\Delta t}{2}) - \hat{h}(t+\frac{\Delta t}{2}) \qquad (8\text{-}28)$$

An approximate, scaled (by C_{vd3}) integral of the altitude error, $\hat{\delta}\alpha$, is computed and applied at the acceleration (actually incremental velocity) level when a third order loop is employed. Its effect is to provide steady state compensation for vertical accelerometer or altimeter bias errors. It is mechanized as

$$\hat{\delta}\alpha(t+\frac{\Delta t}{2}) = \hat{\delta}\alpha(t-\frac{\Delta t}{2}) + C_{vd3}\,\hat{\delta}h(t+\frac{\Delta t}{2})\,\Delta t \qquad (8\text{-}29)$$

It is applied to the vertical velocity as indicated in 8-19.

10) <u>Extract Latitude, Longitude and Wander Angle from A Matrix</u>

The procedure is outlined in Appendix B.

$$\cos L(t) = \left[A_{11}^2(t) + A_{21}^2(t) \right]^{1/2} \tag{8-30}$$

If cos L(t) = 0, latitude is $\pm 90°$ and longitude and wander angle are indeterminate and previous values are output.

$$* \quad L(t) = \tan^{-1}\left[\frac{A_{31}(t)}{\cos L(t)}\right] \tag{8-31}$$

$$* \quad \ell(t) = \tan^{-1}\left[\frac{A_{21}(t)}{A_{11}(t)}\right] \tag{8-32}$$

$$\alpha(t) = \tan^{-1}\left[\frac{A_{32}(t)}{A_{33}(t)}\right] \tag{8-33}$$

During the succeeding operations and the next computational cycle, the extrapolated values of the sine and cosine of the wander angle will be required. They are obtained from

$$\delta\alpha \underset{=}{\Delta} \alpha(t) - \alpha(t-\Delta t)$$

$$S\hat{\alpha}(t + \frac{\Delta t}{2}) \doteq S\alpha(t) + \frac{\delta\alpha}{2} C\alpha(t) \tag{8-34}$$

$$C\hat{\alpha}(t + \frac{\Delta t}{2}) \doteq C\alpha(t) + \frac{\delta\alpha}{2} S\alpha(t)$$

12) <u>Transform Computational Frame Velocities to ENU Frame</u>

The equation to be mechanized is

$$\underline{v}^{\ell}(t) = C_{c'}^{\ell}(t)\ \underline{v}^{c'}(t) \tag{8-35}$$

$$= C_{\ell'}^{\ell}(t)\ C_{c'}^{\ell'}(t)\ \underline{v}^{c'}(t)$$

---

*Normal navigation outputs.

This is the inverse of equations 8-2 and 8-2a. Both transformations are given in Appendix B, hence

$$\underline{v}^{\ell}(t) = X(\pi/2)\ Z(\pi/2)\ \tilde{X}(\alpha)\ \underline{v}^{c'}(t)$$

or

$$* \begin{bmatrix} v\ {}^{q}t0 \\ v\ (t) \\ v\ (t) \end{bmatrix} = \begin{bmatrix} 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^{c'}(t) \\ v_2^{c'}(t) \\ v_3^{c'}(t) \end{bmatrix}$$

The value of α above is $\alpha(t)$.

13) Compute Gravity, Extrapolated to Mid-Computation Cycle

The components of gravity in the computational frame at the midpoint of the next computational cycle, $g^{c'}(t + \frac{\Delta t}{2})$, are required for the velocity update on the next pass. Subroutine GRAV is entered with arguments $\hat{\lambda}(t + \frac{\Delta t}{2})$ and $\hat{h}(t + \frac{\Delta t}{2})$. Equations 9-16 a, b, and c are mechanized returning $g^{c'}(t + \frac{\Delta t}{2})$ and $\sin^2 \hat{L}(t + \frac{\Delta t}{2})$.

14) Compute Extrapolated Angular Velocity

The angular velocity of the computational frame with respect to the earth fixed frame $\underline{\rho}(t + \frac{\Delta t}{2})$, is computed for use in the upgraded estimates of the Coriolis corrections, which follow.

The arguments supplied to subroutine ANGVEL are

$$\hat{v}^{c'}(t + \frac{\Delta t}{2})$$

$$c\hat{\alpha}(t + \frac{\Delta t}{2})$$

$$s\hat{\alpha}(t + \frac{\Delta t}{2})$$

$$s^2\hat{L}(t + \frac{\Delta t}{2})$$

$$\hat{h}(t + \frac{\Delta t}{2})$$

ANGVEL implements Equations 8-3 through 8-7.

15) Compute Extrapolated Coriolis Corrections

The Coriolis corrections for use in the next velocity update are computed in subroutine TORCOR.

The arguments are:

$$\hat{A}(t + \frac{\Delta t}{2})$$

$$\hat{\underline{\rho}}(t + \frac{\Delta t}{2})$$

$$\frac{\hat{\underline{v}}^{c'}}{\Delta t}(t + \frac{\Delta t}{2})$$

The subroutine mechanizes Equations 8-8 through 8-11, and returns $(\underline{\omega}_{cor} \times \hat{\underline{v}}^{c'})\ \Delta t$ at $(t + \frac{\Delta t}{2})$, where $\underline{\omega}$ cor is given by Equation 8-10.

16) Reset Incremental Velocity Input

The $\underline{\Delta v}^{i'}(t)$ received from the velocity-attitude algorithm is reset to zero and passed back to the V-A module.

This anticipates the case, which often occurs in practice, wherein the velocity-attitude algorithm is exercised several times for each pass through the navigation algorithm.

17) <u>Compute "Attitude" Matrix and Extract Attitude Angles</u>

Using the DCM, or $C_b^{e'}$, from the velocity attitude algorithm, and $\tilde{A}$ or $C_{e'}^{c'}$, from the navigation algorithm, the "attitude" matrix, $C_b^{c'}$ is formed:

$$C_b^{c'}(t) = C_{e'}^{c'}(t)\, C_b^{e'}(t) \tag{8-37}$$

Equation 8-37 is developed and interpreted in Appendix B, where the final result is

$$C_b^{c'} = Y(\pi/2)\, \tilde{Z}(\psi w) \cdot \tilde{Y}(\theta)\, \tilde{X}(\phi)\, X(\pi) \tag{8-38}$$

$$= \left[\begin{array}{c|c|c} s\theta & c\theta s\phi & c\theta c\phi \\ s\psi_w c\theta & -c\psi_w c\phi - s\psi_w s\theta s\phi & c\psi_w s\phi - s\psi_w s\theta c\phi \\ c\psi_w c\theta & s\psi_w c\phi - c\psi_w s\theta s\phi & -s\psi_w s\phi - c\psi_w s\theta c\phi \end{array}\right]$$

$$\begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_9 \end{bmatrix} = \text{DTEM in program mnemonics}$$

From the above (the elements of the first row and column of $C_b^{c'}$) the "attitude" angles are extracted, viz.

$$\cos\theta = \left(C_2^2 + C_3^2\right)^{1/2} \tag{8-39}$$

If $\cos\theta$ is 0 then $\theta = \pm 90°$ and ψ_w and ϕ are indeterminate and the previous values are output. Otherwise,

$$\psi_w = \tan^{-1}\left(\frac{C_4}{C_7}\right) \tag{8-40}$$

$$^{*}\quad \phi = \tan^{-1}\left(\frac{C_2}{C_3}\right) \tag{8-41}$$

$$^{*}\quad \theta = \tan^{-1}\left(\frac{C_1}{\cos\theta}\right) \tag{8-42}$$

---

*Normal navigation outputs

Finally, the true heading, ψ, is computed from

$$^{*}\ \psi = \psi_w - \alpha \tag{8-43}$$

where α is obtained from equation 8-33 and the argument, t, has been omitted for brevity in equations 8-38 through 8-43.

The $\underline{\Delta v}^{i'}(t)$ received from the velocity attitude algorithm is reset to zero and passed back to the V-A module. The module operating time is stored for output, then incremented for the next pass.

The nine stored (*) quantities and time are output to the evaluation module, and the navigation module awaits the next pass.

---

*Normal navigation outputs.

SECTION 9

EVALUATION OF ERRORS

The evaluation Module (EVΔ) is the final, minor module of the INSS. Its functions are

(a) To store the position, velocity, and attitude from the trajectory module at the time of each navigation and attitude computation cycles.

(b) To store the position, velocity, and attitude from the navigation module after every navigation computation cycle.

(c) To output the trajectory data (50 points at a time) to the trajectory data and output the resulting differences or errors (50 points at a time) to the line printer and plot file.

The net result is a time history of the nominal trajectory, i.e., without any perturbations from the environment module, and an instantaneous "error" time history. As presently structured, "errors" include the effects of the linear and angular vibration on the simulated strapdown navigator, but not on the nominal trajectory. This is an obvious deficiency of the present program since the total effect of its vibration environment should be reflected in the position, velocity, and attitude, comprising the nominal trajectory, as reported by EVL.

REFERENCES

1) Seppelin, T.O., The Department of Defense World Geodetic System, 1972, Defense Mapping Agency, May 1974.

2) Musick, S.T., PROFGEN - A Computer Program for Generating Flight Profiles, Technical Memorandum 76-3, Air Force Avionics Laboratory, WPAFB, Ohio, March 1976.

3) McKern, R.A., A Study of Transformation Algorithms for Use in a Digital Computer, T-493, (Master's Thesis), MIT, Cambridge, Mass., January 1968.

4) Britting, K.R., Inertial Navigation Systems Analysis, Wiley-Interscience, New, York, N.Y., 1971.

5) Sciegienny, J., Nurse R., et.al., Inertial Navigation Sytem Standardized Software Development, Final Technical Report, R-977, C.S. Draper Lab., Cambridge, Mass., June 1976.

APPENDIX A

ROTATION MATRICES

A.1 Introduction

A simple, compact, unambiguous notation describing transformation from one orthogonal axis frame to another in terms of one or more single-axis rotations is presented in this section.

The notation used in describing the vectors and the coordinates transformations is based on notation used by Britting.[4] The definitions and the notations presented in this Appendix are employed in Appendix B to describe the coordinate frames and transformations occurring in the INSS.

A.2 Single-Axis Rotations

A coordinate frame, i, is defined by specifying three orthogonal unit vectors $\hat{X}_i$, $\hat{Y}_i$, $\hat{Z}_i$, which form a right-handed system, i.e.,

$$\hat{X}_i \times \hat{Y}_i = \hat{Z}_i, \; \hat{Y}_i \times \hat{Z}_i = \hat{X}_i, \; \hat{Z}_i \times \hat{X}_i = \hat{Y}_i$$

$$X_i \cdot Y_i = 0, \; Y_i \cdot Z_i = 0, \; Z_i \cdot X_i = 0$$

where "×" denotes the vector product and "·" the scalar product of two vectors.

The "i" is the shorthand name of the frame, and is one or more numbers or letters, or some combination thereof. The practice of referring to the unit vectors defining a frame as $\hat{i}$, $\hat{j}$, $\hat{k}$; $\hat{u}$, $\hat{v}$, $\hat{w}$; $\hat{M}$, $\hat{N}$, $\hat{O}$;

$\hat{X}$, $\hat{Y}$, $\hat{Z}$; etc. will not be used because there are too many frames and one would soon run out of triples of letters for which the sequence is obvious. Further, the existence of an X-axis, a Y-axis, and a Z-axis in each frame is indispensable to the concept of single-axis rotations.

Consider two Cartesian coordinate frames with a common origin. Denote the first frame by i and the second by j. Then the unit vectors defining the two frames are X_i, Y_i, Z_i, and X_j, Y_j, Z_j. If one axis of one frame coincides with the corresponding axis of the other frame, i.e., if

$$\hat{X}_i = \hat{X}_j \text{ or } \hat{Y}_i = \hat{Y}_j \text{ or } \hat{Z}_i = \hat{Z}_j$$

then the transformation from one frame to the other is at most a single-axis rotation. The axis which is common to the two frames is the axis of rotation and tne angle between the corresponding non common axes is the angle of rotation or argument. The sign of the angle of rotation is positive if the non common axes of the first ("from") frame move in a clockwise direction to bring themselves into coincidence with the corresponding axes of the second ("to" frame, looking in the positive direction along the common axis (right-hand screw rule).

(1) <u>X-axis Rotation, X (α)</u>

If $\hat{X}_i = \hat{X}_j$, the transformation is an X-axis rotation; and if the angle between Y_i and $\hat{Y}_j$, and $\hat{Z}_i$ and $\hat{Z}_j$, is α and if α is positive, then transformation from i to j is written as X(α). This is illustrated in Figure A-1.

The equations relating the unit vectors of the j frame axes to the unit vectors of the i frame axes are:

$$\hat{X}_j = \hat{X}_i$$
$$\hat{Y}_j = \cos\alpha\,\hat{Y}_i + \sin\alpha\,\hat{Z}_i$$
$$\hat{Z}_j = -\sin\alpha\,\hat{Y}_i + \cos\alpha\,\hat{Z}_i$$

or in matrix form

$$\begin{bmatrix} \hat{X}_j \\ \hat{Y}_j \\ \hat{Z}_j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \hat{X}_i \\ \hat{Y}_i \\ \hat{Z}_i \end{bmatrix}$$

$$X(\alpha) \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

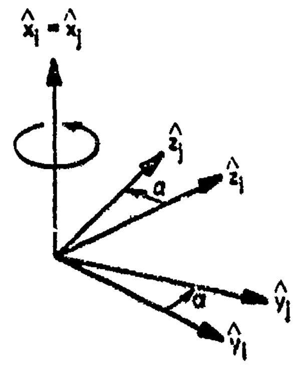


Figure A-1. Rotation about X-axis.

In this particular case, the roation from i to j, denoted by C_i^j is $X(\alpha)$. It is often helpful to represent a number of single-axis rotations in a signal flow graph form as shown in Figure A-2.

$$\begin{tikzcd} i \arrow[r, "\alpha", "X"'] & j & = & i & j \arrow[l, "-\alpha"', "X"] \end{tikzcd}$$

"from" frame   axis of rotation   "to" frame

Figure A-2. Rotation about X-axis flow diagram.

The arrowhead indicates the "to" frame. Changing the direction of the arrow corresponds to changing the sign of the argument or transposing the rotation matrix.

(2) Y-axis Rotation, $Y(\beta)$

Similarly, if $Y_j = Y_k$ the transformation from the j frame to the k frame is a Y-axis rotation. For a positive argument, β, this is written as $Y(\beta)$ and is illustrated in Figure A-3.

$$Y(\beta) \triangleq \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

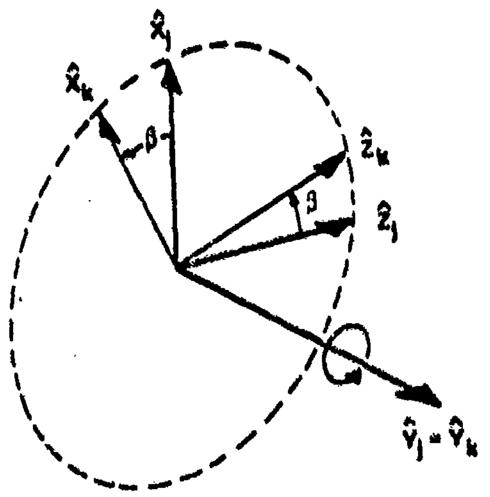


Figure A-3. Rotation about Y-axis.

In the signal flow form we have for C_j^k (see Figure A-4).

$$\begin{tikzcd} j \arrow[r, "\beta", "Y"'] & k \arrow[r, phantom, "="] & j & k \arrow[l, "-\beta"', "Y"] \end{tikzcd}$$

Figure A-4. Rotation about Y-axis flow diagram.

(3) Z-axis Rotation, Z( )

Finally, if $Z_k = Z_\ell$, the transformation from the k frame to the ℓ frame, C_k^ℓ , for a positive argument, γ, is written as $Z(\gamma)$ and is illustrated in Figure A-5.

$$Z(\gamma) \triangleq \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

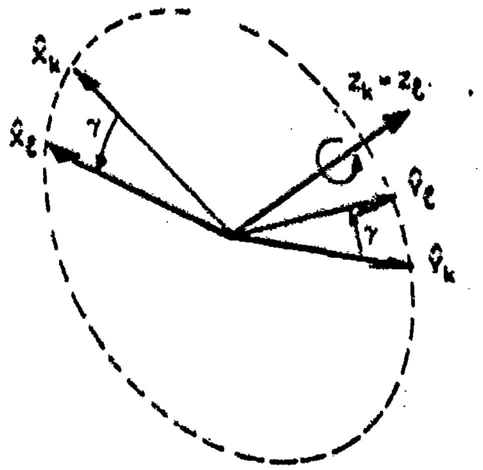


Figure A-5. Rotation about Z-axis.

In signal flow form we have for C_i^ℓ (see Figure A-6).

$$\begin{tikzcd} k \arrow[r, "\gamma", "Z"'] & \ell \end{tikzcd} = \begin{tikzcd} k & \ell \arrow[l, "-\gamma"', "Z"] \end{tikzcd}$$

Figure A-6. Rotation about Z-axis flow diagram.

A.3 The General Rotation

If the three single-axis rotations are cascaded, the transformation from the i frame to the ℓ frame, C_i^ℓ, may be written as

$$\begin{aligned} C_i^\ell &= C_k^\ell \, C_j^k \, C_i^j \\ &= Z(\gamma) \; Y(\beta) \; X(\alpha) \end{aligned}$$

Note, that the subscripts cancel the superscripts from lower left to upper right, leaving only the lower right subscript ("from") and the upper left superscript ("to"). This latter notation appears in Britting[4] The vector is designated by a small letter with one or more subscripts, if required. The frame in which a vector is defined is denoted by a superscript and transforming a vector $\underline{v}$ from the i frame to the ℓ frame is expressed as

$$\underline{v}^\ell = C_i^\ell \, \underline{v}^i$$

For a vector with notation employing two subscripts, such as the angular velocity of the ith frame with respect to the jth frame, expressed in the kth frame is expressed by $\underline{\omega}_{j,i}^k$.

Several of these general rotation matrices occur in the simulation and mechanization of a strapdown INS. The required transformations are presented in Appendix B.

A prerequisite to establishing a simple, consistent set of single-axis rotations relating the frames of interest is to define the coordinate axes of one frame, and then relate the others to it via the appropriate rotations.

APPENDIX B

COORDINATE FRAMES AND TRANSFORMATIONS AND THEIR USE IN THE INSS

B.1 Introduction

In any inertial navigation which provides position and velocity with respect to the earth in a local geodetic vertical north pointing (LVN) frame and the attitude (roll, pitch, and heading) of the vehicle with respect to the same LVN frame, there are generally four fundamental coordinate frames of interest, namely:

1. The Platform Frame
   - Defined by the input axes of the inertial sensors.

2. The Body (or Vehicle) Frame
   - Defined by the longitudal, lateral and normal axes of the vehicle.

3. The Local Vertical North Frame
   - Defined by the local (geodetic) vertical, east (or west) and north (or south) axes.
   - Fixed with respect to the earth at a point below (or above) the vehicle.

4. The Inertial Reference Frame

- Defined by the earth's polar axis and the equatorial plane (at the start of navigation).
- Nonrotating with respect to the celestial sphere.

Only the first of these fundamental coordinate frames, (1), is a function of the configuration and mechanization of the particular terrestrial inertial navigator. In fact, the three classical mechanizations of an INS are based on the platform frame coinciding with one of the three remaining fundamental coordinate frames. Thus, if the platform frame coincides with the body frame, we have a strapdown system; if the platform frame coincides with the local vertical north frame, we have a local vertical system; if the platform frame coincides with an inertial frame, we have a space stable system. The latter two types imply stabilization of the platform or, in these cases, the stable element by gimbals and control loops driven by gyro and other signals; the strapdown system mounts the instrument cluster directly to the body. It is this type of inertial measuring unit (IMU) which is considered in this report.

B.2 Digression

An added complication has crept into the mechanization of strapdown systems due to the fact that they were proceded chronologically by local vertical systems. While local vertical systems work very well over most of the earth's surface, both computational and physical problems arise in the vicinity of the earth's poles. The computational problem occurs in the calculation of the longitude rate which is of the form $V_E/(R \cos L)$, where V_E is the east velocity with respect to the earth, R is the radius of the earth, and cos L is the cosine of the latitude. As the latitude approaches 90 degrees (at the poles), cos L approaches zero and the longitude computation blows up. The physical limitation arises in the computation of the azimuth gyro torquing command, i.e., the rate at which the platform must be rotated about a vertical axis to maintain one of its level axes north. The torquing rate is of the form $[(V_E/R) \tan L + \omega_{ie} \sin L]$, where ω_{ie} is the sidereal earth's rate, and tan L and sin L are the tangent and sine of the latitude. Again, as the latitude approaches 90 degrees, the torquing rate may approach infinity. To circumvent these singularities, the local vertical wander azimuth (LVWA) mechanization was developed, wherein the "azimuth" gyro torquing was reduced or eliminated by incorporating its effects in a "wander" angle which becomes one of the Euler angles in a direction cosine matrix (d.c.m.). Since a d.c.m. never becomes singular, the extraction of latitude, longitude, and wander angle from the Euler angles of the d.c.m. avoids the problem of the north-slaved, local vertical mechanization. However, both the longitude and wander angles are indeterminate at the poles.

The computational and physical limitations of a north-slaved, local vertical system need not arise in strapdown (or space stable) system, since the gyros are not torqued to maintain a preferred orientation with respect to the earth. The IMU of a strapdown system outputs incremental velocities and angles—the integrals of specific force and angular velocity—in the body (platform) frame. If the incremental velocity outputs are transformed to the inertial frame and integrated using Cartesian coordinates, the position and velocity may be expressed in geographic coordinates without resorting to the introduction of a wander angle.

However, in the present simulation, the use of a LVWA computational frame was specified by customer, so the incremental velocities in the inertial frame are transformed to LVWA computational frame, where the navigation algorithm appears the same as it would for a gimbaled, local vertical IMU—except that the gyro torquing signals become the matrix torquing signals for the inertial-to-LVWA transformation.

The purpose of this digression has been the explanation (if not justification) of the introduction of an additional coordinate frame, the LVWA computational frame. This frame coincides with the LVN frame when the wander angle, α, is zero. When the nominally "north" axis (for $\alpha = 0$) is rotated to the west of true north, the wander angle is positive.

B.3 The Simplest Case

If one were seeking the simplest mechanization of a strapdown system for terrestrial navigation, there would be only three coordinate frames of interest. They are as listed:

1. The platform (P), and body (b) (or vehicle) frame.
2. The local vertical north frame.
3. The inertial reference frame.

In Figure B-1, a gross "circle diagram" illustrates the three coordinate frames of interest (denoted by the small circles), and the intervening transformation (denoted by the rectangular boxes). The circle diagram is a pictorial representation of the fact that the product of the rotations around a closed path in the same sense is identity or

$$C^{b}_{l'} C^{l'}_{i'} C^{i'}_{b} = I \qquad \text{(B-1)}$$

With the direction of the rotations, as shown by the arrowheads in Figure B-1, which corresponds to premultiplying both sides of Eq. B-1 by $C^{l'}_{b'}$, the attitude matrix, we have

$$C^{l'}_{b'} C^{b'}_{l'} C^{l'}_{i'} C^{i'}_{b} = C^{l'}_{b} I \qquad \text{(B-2)}$$

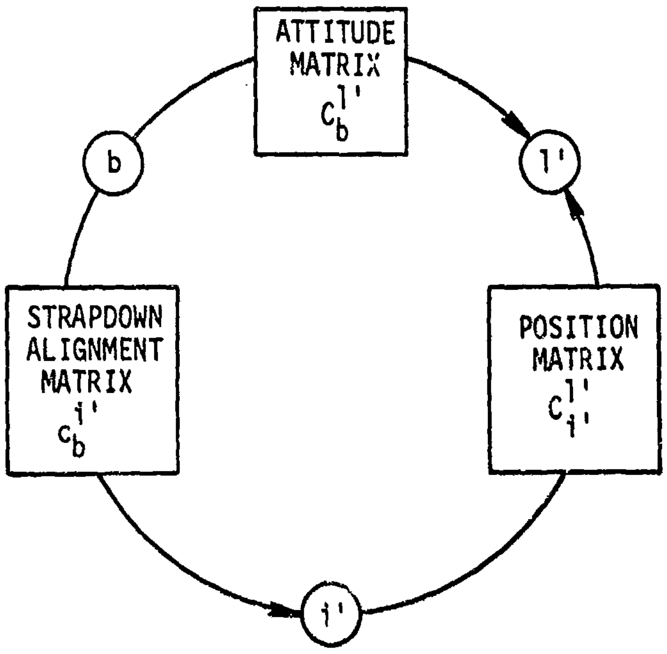

ATTITUDE MATRIX $C_b^{l'}$
b
l'
STRAPDOWN ALIGNMENT MATRIX $C_b^{i'}$
POSITION MATRIX $C_{i'}^{l'}$
i'

Figure B-1. Transformations required for strapdown (SD INS computations - simplest case).

cc

$$C_b^{l'} = C_{i'}^{l'} C_b^{i'} \qquad \text{(B-3)}$$

where Britting's notation(4) is used throughout. Any orthogonal transformation, C_j^k, corresponds to one or the product of two or more single-axis rotations, as indicated in Appendix A.

To maintain the simplest relationships among the fundamental coordinate frames, requires that, when the angles specified by the rotations are all zero, the coordinate frames are coincident. This is accomplished by specifying the coordinate axes of one frame and relating it to the others by the simplest sequence of single-axis rotations.

If the Z axis of the inertial reference frame is defined to lie along the earth's polar axis (EPA) with the positive end at the north pole, and the X axis is defined by the intersection of the equatorial plane and the meridian plane (observer's or Greenwich at time $t = 0$), then the Y axis lies in the equatorial plane, 90 degrees east of observer's or Greenwich at $t = 0$. This defines an earth-centered inertial (ECI) reference frame for all $t \geq 0$. If the first definition is used, then the longitude computed is the change in observer's longitude since $t = 0$, and the observer's initial longitude relative to the Greenwich meridian, ℓ_o, is simply an additive constant. Positive longitude, or change in longitude , is to the east along the equator, i.e., it is a positive rotation about the EPA. The rotation of the earth with respect to the ECI frame, $\omega_{ie}t$, is about the same axis and in the same sense as longitude.

Then, to define the location of a point on the earth, P, the observer's location at time t, a rotation about an easterly axis parallel to the equatorial plane, through the geodetic latitude, L, is required. Since latitude is defined to be positive in the northern hemisphere, the sense of the rotation is negative in the northern hemisphere.

In the compact notation of Appendix A, the transformation from the inertial reference frame to the LVN frame (observer's position with respect to the earth at time t), $C_{i'}^{l'}$, is expressed as the product of two single-axis rotations:

$$C_{i'}^{\ell'} = Y(-L)Z(\ell + \omega_{ie}t) \qquad \text{(B-4)}$$

The X, Y, Z axes of the LVN (ℓ') frame previously defined are Up, East, North, respectively.

If the aircraft were headed north and the wings and nose were level, all the attitude angles would be zero and the body frame would coincide with the LVN frame. However, if the aircraft performs a climbing, right-hand turn, all three "attitude" angles (roll, ϕ; pitch, θ; heading, ψ) will assume positive values. The simplest transformation from the UEN (LVN) frame to a body-fixed frame involves three successive single-axis rotations through the heading, pitch, and roll. They are as follows:

1. The first is a negative rotation about the X or "Up" axis of the LVN frame through the heading angle, ψ; i.e., when ψ goes positive, the nose of the aircraft is rotating from north to the east. This rotation is denoted by $X(-\psi) = \tilde{X}(\psi)$, where the tilde (~) denotes the transpose.

2. The second is a positive rotation about the Y axis of the intermediate frame (which is displaced from the UEN frame by $X(-\psi)$; i.e., the Y axis is East if $\psi = 0$) through the pitch angle, θ. When the pitch angle goes positive, the nose is elevated above the horizontal plane. This rotation is denoted by $Y(\theta)$.

3. The third is a positive rotation about the forward-pointing, longitudinal (Z) axis of the aircraft through the roll angle, ϕ. Roll is positive when the right wing dips below the horizontal plane. This rotation is denoted by $Z(\phi)$.

The complete UEN(LVN) to body transformation, $C_{\ell'}^{b'}$, is given by

$$C_{\ell'}^{b'} = Z(\phi)Y(\theta)\;X(-\psi) \qquad \text{(B-5)}$$

Note that the X, Y, Z, "body" axes in this case are "Up" through the the canopy, and through the right wing, and forward through the nose, respectively.

The attitude angles as previously defined in Eq. (B-5) correspond to those obtained from perfect angle encoders on an ideal, three-axis, local vertical, north-slaved IMU, where the gimballing order, from the outside in is roll, pitch, and azimuth. This gimbal arrangement is used on cruise vehicles, where the pitch angle does not approach 90 degrees. On tactical aircraft with all-attitude capabilities, a redundant, inner-roll gimbal is added frequently with limited freedom—which drives the outer-roll gimbal so as to maintain the inner-roll gimbal orthogonal to the pitch gimbal (except during certain maneuvers). This gimbal arrangement and the resulting "attitude" matrix is illustrated in Figure A-9 in Section A.2 of Appendix A, Volume II.

In a strapdown system, the gyros are usually mounted with their input axes along (or parallel to) the aircraft body axes. The gyros ideally sense the angular velocity of the body frame with respect to the inertial frame, expressed in the body frame. The outputs of the gyros represent the integrals of these angular velocities over a short interval of time. So if the initial orientation of the body with respect to the inertial frame, $C_{b'}^{i'}(0)$ or $C_{i'}^{b'}(0)$ inertial-to-body, is known and one properly keeps track of the changes in orientation with time, then $C_{b'}^{i'}(t)$ is always known.

Now, premultiplying Eq. (B-4) by Eq. (B-5) we have

$$\begin{aligned} C_{i'}^{b'} &= C_{\ell'}^{b'} C_{i'}^{\ell'} \\ &= Z(\phi)Y(\theta)X(-\psi)Y(-L)Z(\ell + \omega_{ie}t) \end{aligned} \qquad \text{(B-6)}$$

Actually, the transpose of the matrix given in Eq. (B-6) or $C_{b'}^{i'}$ is computed in the "velocity-attitude" algorithm which computes (or can compute) the inertial to local vertical transformation, $C_{i'}^{\ell'}$. Then by premultiplying the former by the latter, we obtain the "attitude" matrix, i.e.,

$$C_{b'}^{\ell'} = C_{i'}^{\ell'} C_{b'}^{i'} \qquad \text{(B-7)}$$

Since the form of Eq. (B-7) is known from Eq. (B-5), the attitude angles, ψ, θ, ϕ, may be extracted, as shown in Section A.4.4 of Appendix A, Volume II. This simple case is illustrated in Figure B-2).

B.4 The Actual Case

The coordinate frames and transformations actually employed in the INSS are considerably more complicated than the simple case just presented. The complication arises due to the introduction of seven additional single-axis rotations required to form the body-to-computational frame transformation, $C_b^{c'}$. Two of these additional rotations occur due to the use of a LVWA computational frame, with the introduction of the wander angle, α, which appears twice in the sequence, while the remaining five are fixed rotations which occur in ones or twos to accommodate the various definitions of local level or inertial frames which appear in the different modules. The overall nominal body-to-computational frame circle diagram is shown in Figure B-3.

Before discussing, this circle diagram in detail it may be instructive to examine $C_b^{c'}$ when all the variable angles, ϕ, θ, ψ_w, α, L, ℓ, and $\omega_{ie}t$ are zero, remembering that in the simple case all the coordinate frames would be coincident. This situation is depicted in Figure B-4 where all the remaining six frames are regarded as LVN frames. Under these circumstances five different LVN frames appear since the north, west, up frame occurs twice.

In the Figure B-4, $C_b^{c'}$, with all variable angles zero is given by

$$C_b^{c'}\Big|_{\substack{\text{all variable}\\ \text{angles zero}}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = Z(\pi)Y(\pi/2)$$

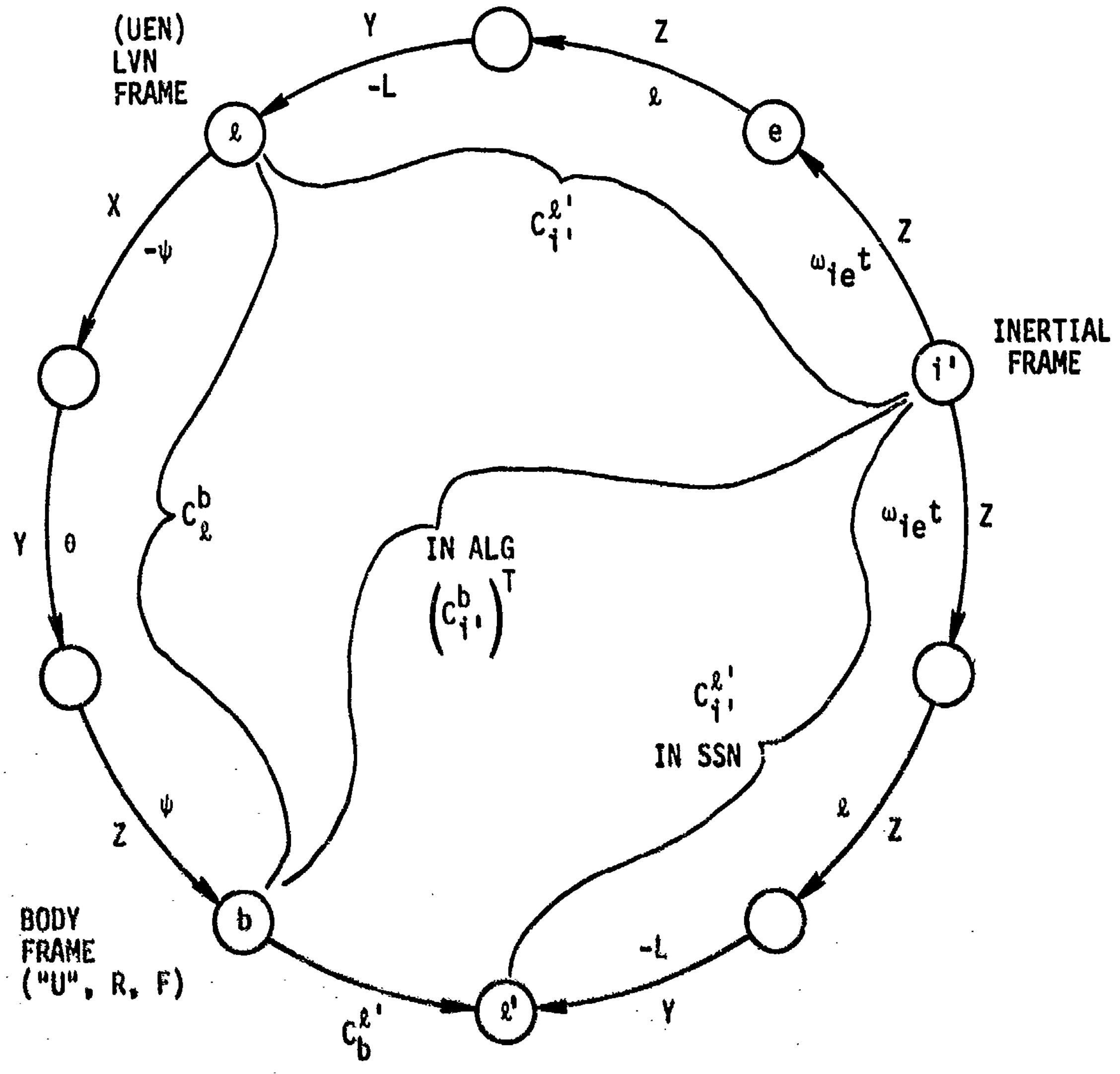


Figure B-2. Simple body-to-LVN frame, C_b^{ℓ}, (attitude matrix).

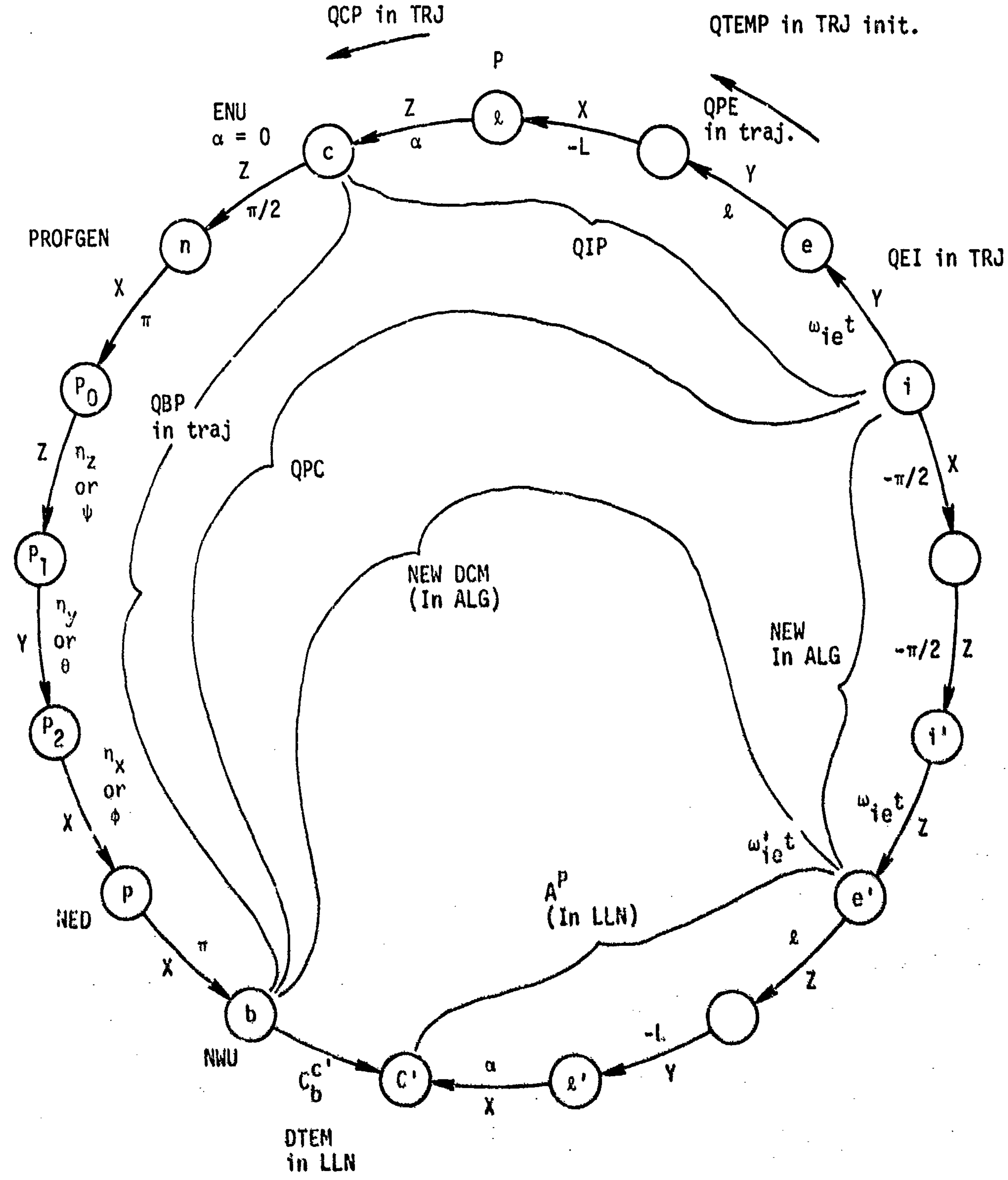
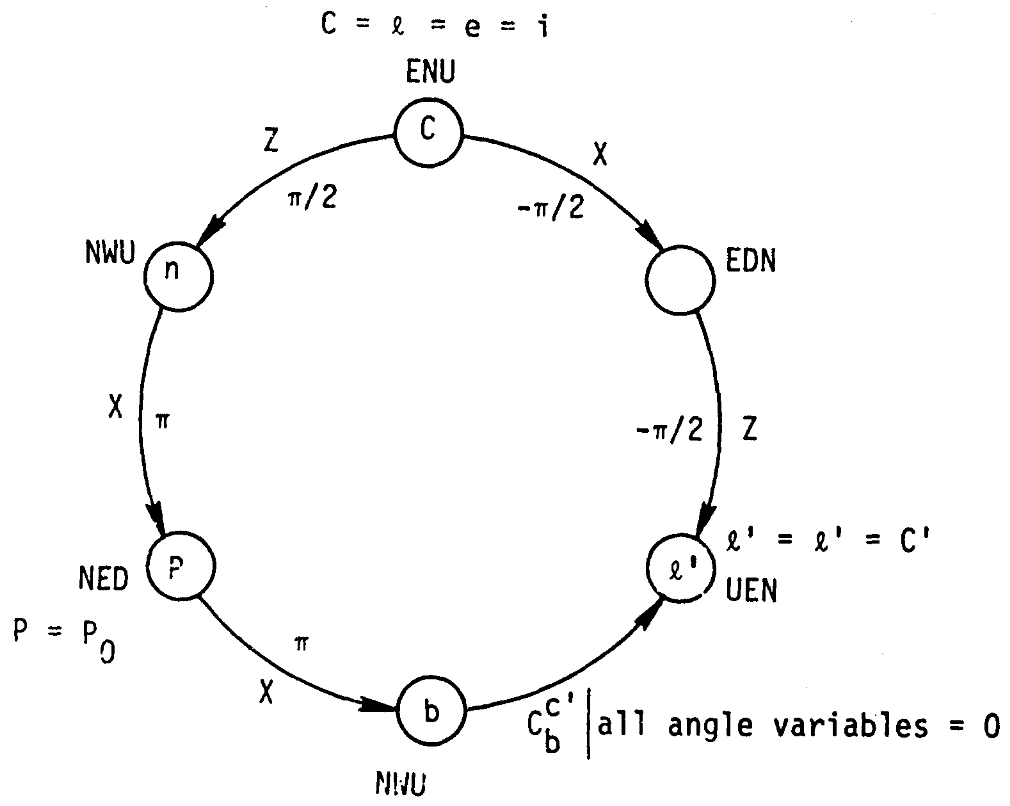


Figure B-3. INSS body-to-LVWA computational frame, $C_b^{c'}$ (circle diagram).



Vol. II, App. B

Figure B-4. $C_b^{c'}$ in INSS with all variable angles zero.

Hence, it requires a total of seven fixed, single-axis rotations to complete the trip around the circle diagram when all the variable angles are zero. With a little planning, all these rotations could have been eliminated, as illustrated by the "Simplest Case"—or as used in Reference 2.

In practice, it is common to encounter different definitions of the same kind of coordinate frames, although seldom as extreme as shown in Figure B-3.

B.5 The LVWA (Computational)-to-Body Frame Transformation, C_c^b or QBC

This transformation relates a wander azimuth, local vertical, computational frame (corresponding to the stable element of a perfect LVWA navigator) to the body frame. It is used in the interim trajectory module as part of the transformations required to generate the incremental velocities and angles which would be output by an ideal strapdown IMU. It is expressed as the product of six single-axis rotations—three fixed, and three variable:

$$C_c^b \triangleq X(\pi)X(\phi)Y(\theta)Z(\psi_w)X(\pi)Z(\pi/2)$$

where

ϕ = the roll (Euler) angle.

θ = the pitch (Euler) angle.

ψ_w = the yaw (Euler) angle of the wander azimuth platform.

Performing the indicated operations, we have

$$C_c^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\phi \\ 0 & 1 & 0 \\ s\phi & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\psi_w & s\psi_w & 0 \\ -s\psi_w & c\psi_w & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c} c\theta\ s\psi_w & c\theta\ c\psi_w & s\theta \\ -s\phi\ s\theta\ s\psi_w - c\phi\ c\psi_w & -s\phi\ s\theta\ c\psi_w + c\phi\ s\psi_w & s\phi\ c\theta \\ -c\phi\ s\theta\ s\psi_w - s\phi\ s\psi_w & -c\phi\ s\theta\ c\psi_w + s\phi\ s\psi_w & c\phi\ c\theta \end{array}\right]$$

where

$c \triangleq \cos$

$s \triangleq \sin$

It should be noted that 3×3 matrices are handled in the program as nine element vector (9×1), where the e ents are stated rowwise, i.e.

$$C_c^b \text{ or } QBC = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_9 \end{bmatrix}$$

or

$$QBC^T = \{C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9\}$$

B.6 <u>The Local Vertical North (LVN) to Local Vertical Wander Azimuth (LVWA) Transformation, C_ℓ^c or QCP</u>

This transformation relates an ENU (LVN) frame to an ideal LVWA frame. It is a single axis rotation about the Up-axis through the wander angle, α, which is positive when the Y-axis of the LVWA frame is west of north.

$$C_\ell^c = Z(\alpha) = \begin{bmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In the Interim Trajectory Module (ITM) the incremental velocities (integrals of specific force) are generated in an East, North, Up (ENU), Local Vertical, North-pointing (LVN) coordinate frame, then are transformed to the body frame (which would be NWU if all the variable angles were zero). In the notation of the circle diagram, Figure B-3, this transformation is denoted by C_ℓ^b . However, as indicated in the heading, the program mnemonic is QBP (P as is in perfect LVN).

$$C_\ell^b = C_c^b C_\ell^c$$

The corresponding body frame specific force, $\vec{a}^b$, is given by

$$\vec{a}^b = C_\ell^b \vec{a}^\ell$$

B.7 <u>The Earth Fixed to LVN Transformation, C_e^ℓ or QPE</u>

This transformation relates an earth-centered earth-fixed frame with its Z and X axes in the equatorial plane, and its Y-axis along the north earth's polar axis (EPA) to the perfect LVN frame with its X, Y, and Z axes along E, N, U. It is the product of two single-axis rotations, namely,

$$C_e^\ell \overset{\Delta}{=} X(-L)Y(\ell)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cL & -sL \\ 0 & sL & cL \end{bmatrix} \begin{bmatrix} c\ell & 0 & s\ell \\ 0 & 1 & 0 \\ -s\ell & 0 & c\ell \end{bmatrix}$$

$$= \begin{bmatrix} c\ell & 0 & s\ell \\ -sLs\ell & cL & -sLc\ell \\ CLS\ell & SL & CL\ell \end{bmatrix}$$

B.8 "The Inertial" to Earth Fixed Transformation, C_i^{ℓ} or QEI

This is the single-axis transformation that relates "the inertial" frame to the earth-centered, earth-fixed frame. It is a position rotation about the north EPA through an angle which is the product of the earth's sidereal rate, ω_{ie}, and the time in navigation, t. It is given by

$$C_i^{\ell} \triangleq Z(\omega_{ie}t) = \begin{bmatrix} c\omega_{ie}t & s\omega_{ie}t & 0 \\ -s\omega_{ie}t & c\omega_{ie}t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The angular rate sensed by the gyros of a strapdown system is that of the body with respect to the inertial frame expressed in the body frame, $\vec{\omega}_{ib}^{b}$. The output of the gyrcs of the SD IMU is the integral of the aforementioned angular rate, over some time interval, Δt, or

$$\overline{\Delta\theta}_n \triangleq \int_{t_{n-1}}^{t_{n-1}+\Delta t} \vec{\omega}_{ib}^{b}(t)\,dt$$

To evaluate this integral, the product of all the four rotation matrices is required; i.e., the transformation from the inertial frame to the body frame at t_n and t_{n-1}, $C_i^{b(t_n)}$, $C_i^{(t_{n-1})}$, where

$$C_i^b \text{ or QBI} = C_c^b \, C_{\ell}^c \, C_e^{\ell} \, C_i^e$$

and

$$C_b^i = [C_i^b]^T = \tilde{C}_b^i$$

where the tilde (~) denotes the transpose, then the product of the previous matrices

$$C_{b(t_{n-1})}^{b(t_n)} = C_i^{b(t_n)} C_{b(t_{n-1})}^{i}$$

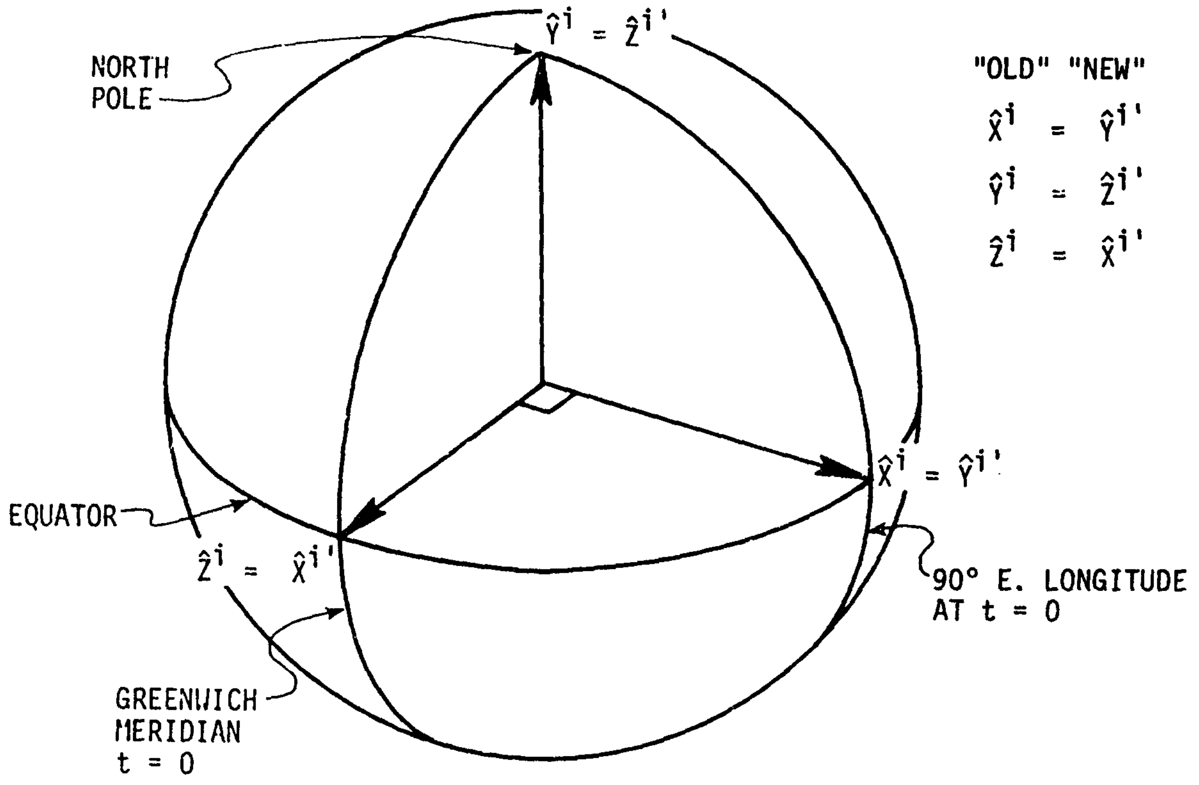
permits the evaluation of $\overline{\Delta\theta}_n$, as indicated in Appendix C.

B.9 The Inertial to "New" Earth-Fixed Transformation, $C_i^{\ell'}$ or Ω

In the earlier discussion of the various LVN frames, which could be considered to be implied by the numerous fixed rotations, it was not specifically noted that some of these frames arose due to two different definitions of the "inertial" frame and the "earth-fixed" frame.

The velocity-attitude algorithm (module) in the INSS is responsible for updating C_b^i on the basis of the $\overline{\Delta\theta}$'s received from the gyro compensation module, so the $\overline{\Delta V}$'s received from the accelerometer compensation module may be transformed from the body frame to the inertial frame used by the navigation algorithm (module). In addition, the velocity-attitude algorithm must pass to the navigation algorithm the body-to-"new"-earth-fixed transformation, $C_b^{\ell'}$, for use by the latter in the attitude computations.

The additional transformation involved in generating $C_b^{e'}$ is $C_i^{e'}$, where e' is the "new" earth-fixed frame, and i' is the "new" inertial frame. These frames, i and e, and i' and e', coincide (in pairs) when $t = 0$. The first pair is based on the X, Y, Z axes of the LVN frame being along East, North, Up, respectively, while the second case assumes the X, Y, Z axes of the LVN frame are along Up, East, North, respectively. The two cases are illustrated in Figure B-5.



NORTH POLE
$\hat{Y}^i = \hat{Z}^{i'}$
"OLD" "NEW"
$\hat{X}^i = \hat{Y}^{i'}$
$\hat{Y}^i = \hat{Z}^{i'}$
$\hat{Z}^i = \hat{X}^{i'}$
$\hat{X}^i = \hat{Y}^{i'}$
EQUATOR
$\hat{Z}^i = \hat{X}^{i'}$
90° E. LONGITUDE AT t = 0
GREENWICH MERIDIAN t = 0

Vol. II, App. B

Figure B-5.

The transformation from the "old" inertial frame, i, to the "new" inertial frame, i', is $C_i^{i'}$ where

$$C_i^{i'} \triangleq \tilde{Z}(-\pi/2)\tilde{X}(-\pi/2)$$

$$= Z(\pi/2)X(\pi/2)$$

From the "new" inertial frame to the "new" earth-fixed frame the transformation, $C_{i'}^{e'}$, is simply

$$C_{i'}^{e'} \triangleq Z(\omega_{ie}t)$$

Finally, the required transformation from the inertial to "new" earth-fixed frame, $C_i^{e'}$, is given by

$$C_i^{'} = C_{i'}^{e'} C_i^{i'}$$

$$= Z(\omega_{ie}t)\tilde{Z}(\pi/2)\tilde{X}(\pi/2)$$

$$= \begin{bmatrix} c\omega_{ie}t & s\omega_{ie}t & 0 \\ -s\omega_{ir}t & c\omega_{ie}t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} s\omega_{ie}t & 0 & c\omega_{ie}t \\ c\omega_{ie}t & 0 & -s\omega_{ie}t \\ 0 & 1 & 0 \end{bmatrix}$$

B.10 The ("New") Earth-Fixed to ("New") Computational (LVWA) Frame Transformation, $C_{e'}^{c'}$ or A^T or A

This transformation (or its transpose) is loosely and generally referred to as "the direction cosine matrix", or d.c.m., in inertial navigation systems employing a LVWA navigation algorithm for the simple reason that it may be the only rotation matrix or d.c.m. occurring in the navigation software for a gimballed, LVWA INS.

The elements of $C_{e'}^{c'}$ are the repository for the earth-relative position (latitude and longitude and wander angle) which are the three Euler angles relating the "new" earth-fixed frame, e', to the "new" computational frame, C'. $C_{e'}^{c'}$ is given by

$$C_{e'}^{c'} \text{ or } \tilde{A} \triangleq X(\alpha)Y(-L)Z(\ell)$$

$$= X(\alpha)\tilde{Y}(L)Z(\ell)$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & s\alpha \\ 0 & -s\alpha & c\alpha \end{bmatrix} \begin{bmatrix} cL & 0 & sL \\ 0 & 1 & 0 \\ -sL & 0 & cL \end{bmatrix} \begin{bmatrix} c\ell & s\ell & 0 \\ -s\ell & c\ell & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c} cLc\ell & cLs\ell & sL \\ -c\alpha s\ell - s\alpha sLc\ell & c\alpha c\ell - s\alpha sLs\ell & s\alpha cL \\ s\alpha s\ell - c\alpha sLc\ell & -s\alpha c\ell - c\alpha sLs\ell & c\alpha cL \end{array}\right]$$

$$= \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{bmatrix}$$

where

α = the wander angle positive to the west of north

L = the geodetic latitude positive in the northern hemisphere

ℓ = the geodetic longitude positive to the east of Greenwich

From the elements of A, position and wander angle are obtained from the following relationships

$$\alpha = \tan^{-1} \frac{A_{32}}{A_{33}}$$

$$\ell = \tan^{-1} \frac{A_{21}}{A_{11}}$$

$$L = \sin^{-1}(A_{31})$$

$$= \tan^{-1}\left(\frac{A_{31}}{\sqrt{A_{11}^2 + A_{21}^2}}\right)$$

$$= \tan^{-1}\left(\frac{A_{31}}{\sqrt{A_{32}^2 + A_{33}^2}}\right)$$

Note that α and ℓ are indeterminate if $A_{11}^2 + A_{21}^2 = A_{32}^2 + A_{33}^2 = 0$, i.e., for CL = 0 or L = ±90°.

B.11 The "Attitude" Matrix or Body to "New" Computational Frame Transformation, $C_b^{c'}$ or DTEM

This transformation relates the body frame to the "new" LVWA computational frame employed in the navigation algorithm. By the appropriate inverse trigonometric relationships, the "attitude" of the vehicle (body) is extracted. "Attitude" included roll, pitch, and yaw. The latter must be corrected by the wander angle to provide the "true" heading.

In the INSS, $C_b^{c'}$ is formed as the product of $C_b^{e'}$ from the velocity-attitude algorithm, and $C_{e'}^{c'}$ from the navigation algorithm, or

$$C_b^{c'} = C_{e'}^{c'} C_b^{e'}$$

This looks simple enough until one reexamines the circle diagram, Figure B-3, and notes that $C_b^{c'}$ involves 16 single-axis rotations and 8 variable rotations through α, L, ℓ, $\omega_{ie}t$ (each of the latter 4 arguments appearing twice). One would expect that the eight aforementioned variable rotations would "cancel" out in pairs, leaving the resultant purely a function of the three attitude angles, ψ_w, θ, and θ, with suitable rearrangement due to the fixed rotations.

In the "simplest" case, shown in Figure B-2, one can see by inspection that this is the case, but in Figure B-3, it is not so obvious that

$$C_b^{c'} = F(\psi_w, \theta, \phi)$$

Letting $\Lambda = \ell + \omega_{ie} t$ and tilde (~) denote transpose $C_b^{c'}$ may be written as

$$C_b^{c'} = C_c^{c'} C_b^c$$

where

$$C_b^c = \tilde{Z}(\pi/2) X(\pi) \tilde{Z}(\psi_w) \tilde{Y}(\theta) \tilde{X}(\phi) X(\pi)$$

and

$$C_c^{c'} = \underbrace{X(\alpha) \underbrace{\tilde{Y}(L) \underbrace{Z(\Lambda) \tilde{Z}(\pi/2) \tilde{X}(\pi/2) \tilde{Y}(\Lambda)}_{} X(L)}_{} \tilde{Z}(\alpha)}_{}$$

It may be shown by expanding $C_c^{c'}$, as indicated by the successive brackets, that only the fixed rotations survive, and

$$C_c^{c'} = \tilde{Z}(\pi/2) \tilde{X}(\pi/2)$$

Hence

$$C_b^{c'} = \tilde{Z}(\pi/2) \tilde{X}(\pi/2) \tilde{Z}(\pi/2) X(\pi) \tilde{Z}(\psi_w) \tilde{Y}(\theta) \tilde{X}(\phi) X(\pi)$$

Expanding the four fixed rotations which are adjacent to one another, $C_{P_o}^{c'}$, we have

$$C_{P_o}^{c'} = \tilde{Z}(\pi/2)\tilde{X}(\pi/2)\tilde{Z}(\pi/2)X(\pi) = Y(\pi/2)$$

The final form of the "attitude" matrix, $C_b^{c'}$, is

$$C_b^{c'} = Y(\pi/2)\tilde{Z}(\psi_w)\tilde{Y}(\theta)\tilde{X}(\phi)X(\pi)$$

$$= \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c\psi_w & -s\psi_w & 0 \\ s\psi_w & c\psi_w & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\psi & -s\psi \\ 0 & s\phi & c\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c} s\theta & c\theta s\phi & c\theta c\phi \\ s\psi_w c\theta & -c\psi_w c\phi - s\psi_w s\theta s\phi & c\psi_w s\phi - s\phi_w s\theta c\phi \\ c\psi_w c\theta & s\psi_w c\phi - c\psi_w s\theta s\phi & -s\psi_w s\phi - c\psi_w s\theta c\phi \end{array}\right]$$

$$= \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \Leftrightarrow \begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_9 \end{bmatrix}$$

The elements of $C_b^{c'}$ are stored as a vector by rows—as indicated by the C_i (not the same C_i as used earlier).

Roll, pitch, and yaw are obtained from the elements of the first row and column of $C_b^{c'}$, namely

$$\psi_w = \tan^{-1}\left(\frac{C_4}{C_7}\right)$$

$$\phi = \tan^{-1}\left(\frac{C_2}{C_3}\right)$$

$$\theta = \sin^{-1}(C_1)$$

$$= \tan^{-1}\left(\frac{C_1}{\sqrt{C_2^2 + C_2^2}}\right)$$

$$= \tan^{-1}\left(\frac{C_1}{\sqrt{C_4^2 + C_7^2}}\right)$$

Note that ψ_w and ϕ are indeterminate if $C_2^2 + C_3^2 = C_4^2 + C_7^2 = 0$; i.e., for $C\theta = 0$ or $\theta = \pm 90°$.

B.12 True Heading

It may be shown that the true heading, ψ, is the difference between the yaw, ψ_w, and the wander angle, α, i.e.,

$$\psi = \psi_w - \alpha$$

In the mechanization, the position and wander angle are extracted before the "attitude" angles, so the current value of α is available when required.

B.13 The "Wander Angle Matrix", or "New" LVWA Computational Frame to LVN (UEN) Frame Transformation, $C_{c'}^{\ell'}$

This single-axis transformation is used to rotate the computational frame velocity to an LVN frame. It is given by

$$C_{c'}^{\ell'} = X(-\alpha) = \tilde{X}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix}$$

and its uses is

$$\underline{v}^{\ell'} = C_{c'}^{\ell'} \underline{v}^{c'}$$

B.14 Other Transformations

Another fixed-angle transformation is employed to express the LVN velocity in the ENU frame, when it is given in the UEN frame. The transformation is $C_{\ell'}^{\ell}$, and is given by

$$C_{\ell'}^{\ell} = X(\pi/2)Z(\pi/2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

and its use is

$$\underline{v}^{\ell} = C_{\ell'}^{\ell} \underline{v}^{\ell'}$$

or

$$\begin{bmatrix} v_x^{\ell} \\ v_y^{\ell} \\ v_z^{\ell} \end{bmatrix} = \begin{bmatrix} v_y^{\ell'} \\ v_z^{\ell'} \\ v_x^{\ell'} \end{bmatrix}$$

This latter form is, of course, the acme of simplicity to code, but it is a frequent source of trouble in interpretation.

If nothing else, the presence of so many redundant transformations points up a very real problem in systems integration, where multiply-defined, similar, coordinate frames are almost certain to occur. The construction and use of the appropriate circle diagram(s) is recommended in these instances.

APPENDIX C

ROTATION MATRIX IN TERMS OF AXIS AND ANGLE OF ROTATION

A direction cosine matrix (or rotation matrix) is unique, its interpretation is variable. As the first name implies, the direction cosine matrix is a 3×3 array, the rows or columns of which are the direction cosines of the axes of one orthogonal frame in another. The rotation matrix may arise in the case where a vector is rotated about a line, i.e., a unit vector with direction cosines in the "body" frame of n_1, n_2, n_3, through an angle θ.

In this case, the rotation matrix from the new body frame to the old (before the rotation) body frame, $R(\theta)$, is given by (3)

$$R = I + N \sin\theta + N^2 (1 - \cos\theta)$$

where

$$N \triangleq \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

and

$$n_1^2 + n_2^2 + n_3^2 = 1$$

Another equivalent expression for R is

$$R = I\cos\theta + \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}\sin\theta + \begin{bmatrix} n_1^2 & n_1n_2 & n_1n_3 \\ n_1n_2 & n_2^2 & n_2n_3 \\ n_1n_3 & n_2n_3 & n_3^2 \end{bmatrix}(1 - \cos\theta)$$

The direction cosines of the axis of rotation n_1, n_2, n_3, may equally well be expressed in terms of the components of the angle of rotation, θ, along the axes of the coordinate frame, as

$$n_1 = \frac{\theta_1}{\theta}$$

$$n_2 = \frac{\theta_2}{\theta}$$

$$n_3 = \frac{\theta_3}{\theta}$$

where

$$\theta = \sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2}$$

In terms of θ and its components, N and N^2 become

$$N = \frac{1}{\theta}\begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}$$

and

$$N^2 = \frac{1}{\theta^2}\begin{bmatrix} -(\theta_2^2 + \theta_3^2) & \theta_1\theta_2 & \theta_1\theta_3 \\ \theta_1\theta_2 & -(\theta_3^2 + \theta_1^2) & \theta_2\theta_3 \\ \theta_1\theta_3 & \theta_2\theta_3 & -(\theta_1^2 + \theta_2^2) \end{bmatrix}$$

and R becomes

$$R = \begin{bmatrix} 1 - \frac{(\theta_2^2 + \theta_3^2)}{\theta^2}(1 - \cos\theta) & -\frac{\theta_3}{\theta}\sin\theta + \frac{\theta_1\theta_2}{\theta^2}(1 - \cos\theta) & \frac{\theta_2}{\theta}\sin\theta + \frac{\theta_3\theta_1}{\theta^2}(1 - \cos\theta) \\ \frac{\theta_3}{\theta}\sin\theta + \frac{\theta_1\theta_2}{\theta^2}(1 - \cos\theta) & 1 - \frac{(\theta_3^2 + \theta_1^2)}{\theta^2}(1 - \cos\theta) & -\frac{\theta_1}{\theta}\sin\theta + \frac{\theta_2\theta_3}{\theta^2}(1 - \cos\theta) \\ -\frac{\theta_2}{\theta}\sin\theta + \frac{\theta_3\theta_1}{\theta^2}(1 - \cos\theta) & \frac{\theta_1}{\theta}\sin\theta + \frac{\theta_2\theta_3}{\theta^2}(1 - \cos\theta) & 1 - \frac{(\theta_1^2 + \theta_2^2)}{\theta^2}(1 - \cos\theta) \end{bmatrix}$$

One should bear in mind that any rotation matrix may be considered to have the form of R (it is not limited to small values of θ) although the simplifications and approximations which are reasonable; and as θ becomes smaller, are evident. If terms on the order of θ^2 are negligible then

$$R\Big|_{\theta^2 \to 0} \approx \begin{bmatrix} 1 & -\theta_3 & \theta_2 \\ \theta_3 & 1 & -\theta_1 \\ -\theta_2 & \theta_1 & 1 \end{bmatrix}$$

since $1 - \cos\theta \to \frac{\theta^2}{2} \to 0$, $\frac{\sin\theta}{\theta} \to 1 - \frac{\theta^2}{6} \to 1$

In general, given R, the angles θ_1, θ_2, θ_3, are extracted as follows

$$\frac{r_{32} - r_{31}}{2} = \theta_1 \frac{\sin\theta}{\theta}$$

$$\frac{r_{13} - r_{31}}{2} = \theta_2 \frac{\sin\theta}{\theta}$$

$$\frac{r_{21} - r_{12}}{2} = \theta_3 \frac{\sin\theta}{\theta}$$

Squaring the above three equations and adding gives

$$(\theta_1^2 + \theta_2^2 + \theta_3^2) \frac{\sin^2\theta}{\theta^2} = \sin^2\theta$$

From this define

$$\sin\theta \triangleq +\sqrt{\sin^2\theta}$$

and hence

$$\theta \triangleq \sin^{-1}(\sin\theta)$$

If an inverse sine routine is not available or convenient, the trace of R_1 on the sum of the diagonal elements is used to define $\cos\theta$, namely,

$$r_{11} + r_{22} + r_{33} = \mathrm{Tr}R = 1 + 2\cos\theta$$

hence,

$$\cos\theta = (\mathrm{Tr}R - 1)/2$$

and the inverse tangent routine may be used to evaluate θ. Note that there is no loss of generality in assuming the positive sign for $\sqrt{\sin^2 \theta}$, since a change of sign here would be reflected in the signs of θ_1, θ_2, and θ_3, effectively flipping the axis of rotation and the sense of the angle of rotation simultaneously.

This interpretation of a rotation matrix is used to generate the incremental angles which would be output by ideal single-degree-of-freedom gyros in a strapdown system, at t_n where the values of C_b^i are given at t_n and $t_n - \Delta t$. In reference (3), the procedure described above is used, but the INSS simply forms the product matrix $C_{b(o)}^{b(n)}$ as

$$C_{b(o)}^{b(n)} = C_i^{b(n)} \left[C_i^{b(o)} \right]^T = C_i^{b(n)} C_{b(o)}^i \rightarrow R^T = \tilde{R}$$

where

$$n \rightarrow n\Delta t, \quad 0 \rightarrow (n - 1)\Delta t$$

Then, three off-diagonal elements are extracted and divided by Δt, and the resultants are called the average angular rates over the nth computation cycle. In addition to the scaling error of $\theta^2/6$, this method also has additive errors in θ_i of $\theta_j \theta_K/2$.

C.1 Relationship Between a Rotation Matrix and a Unit Quaternion

The general rotation matrix, R, shown in expanded form in Eq. (3-), may equally well be expressed exlicitly in terms of the direction cosines of the axis of rotation, n_1, n_2, and n_3, and of trigonometric functions of the half angle of rotation, θ/2.

Using the half-angle trigonometric identities

$$\sin \theta = 2 \sin \frac{\theta}{2} \cos \frac{\theta}{2} = 2 s \frac{\theta}{2} c \frac{\theta}{2}$$

$$1 - \cos \theta = 2 \sin^2 \frac{\theta}{2} = 2 s^2 \frac{\theta}{2}$$

and

$$\theta_1/\theta = n_1,\ \theta_2/\theta = n_2,\ \theta_3/\theta = n_3,\ n_1^2 + n_2^2 + n_3^2 = 1$$

and substituting the above into the expression for R, we have

$$R = \begin{bmatrix} 1 - (n_2^2 + n_3^2) (2\ s^2\ \frac{\theta}{2}) & -n_3 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_1 n_2) 2\ s^2\ \frac{\theta}{2}) & n_2 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_3 n_1 (2\ s^2\ \frac{\theta}{2}) \\ n_3 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_1 n_2 (2\ s^2\ \frac{\theta}{2}) & 1 - (n_3^2 + n_1^2)(2\ s^2\ \frac{\theta}{2}) & -n_1 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_2 n_3 (2\ s^2\ \frac{\theta}{2}) \\ -n_2 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_3 n_1 (2\ s^2\ \frac{\theta}{2}) & n_1 (2\ s\ \frac{\theta}{2}\ c\ \frac{\theta}{2}) + n_2 n_3 (2\ s^2\ \frac{\theta}{2}) & 1 - (n_1^2 + n_2^2)(2\ s^2\ \frac{\theta}{2}) \end{bmatrix}$$

Noting from Reference (3) that the four elements of a unit quaternion, q, may be expressed as

$$q_0 = \cos\frac{\theta}{2} = c\ \frac{\theta}{2}$$

$$q_1 = n_1 \sin\frac{\theta}{2} = n_1\ s\ \frac{\theta}{2}$$

$$q_2 = n_2 \sin\frac{\theta}{2} = n_2\ s\ \frac{\theta}{2}$$

$$q_3 = n_3 \sin\frac{\theta}{2} = n_3\ s\ \frac{\theta}{2}$$

R may be written in terms of the elements of the unit quaternion as

$$R = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_3q_0) & 2(q_3q_1 + q_2q_0) \\ 2(q_3q_0 + q_1q_2) & 1 - 2(q_3^2 + q_1^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_2q_0) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix} = [r_{ij}]$$

Since $q_0^2 = 1 - q_1^2 - q_2^2 - q_3^2$, for a unit quaternion, the elements of the quaternion may be found from the elements of the corresponding rotation matrix, as follows:

$$1 + r_{11} + r_{22} + r_{33} = 1 + \mathrm{Tr}R = 4(1 - q_1^2 - q_2^2 - q_3^2) = 4q_0^2$$

hence

$$q_0 = \sqrt{(1 + \mathrm{Tr}R)/4}$$

and

$$q_1 = (r_{32} - r_{23})/4\ q_0$$

$$q_2 = (r_{13} - r_{31})/4\ q_0$$

$$q_3 = (r_{21} - r_{12})/4\ q_0$$

Both the above conversions are employed in the Velocity Attitude algorithm when a quaternion update algorithm is used.

APPENDIX D

INERTIAL COMPONENT ORIENTATION

D.1 Introduction

Given the body (or vehicle) frame, defined by X_b, Y_b, Z_b, the (nominal) inertial component frames are defined such that the input axes of one inertial component case of each kind lies along (or parallel to) one of the body axes. Hence the X-gyro or accelerometer input axis is nominally aligned parallel to the X-body axis, and similarly for the other inertial components. If the further (and usual) constraint - that the remaining two axes of each inertial component be aligned parallel to the remaining two body axes - is applied, then there are four possible, nominal orientations of each single-degree-of-frredom inertial component. The particular orientations chosen are generally predicated on an attempt to minimize the degradation in component performance in the anticipated vehicle environment, bearing in mind that calibration is usually performed in a one g field. To take one simple example: if a gyro has a large, unstable, g or g^2 dependent, torque coefficient which is forced by acceleration (or acceleration squared) along the spin reference axis (SRA) and the gyro is to be used in an aircraft inertial navigator, then the first choice for the orientation of the gyro SRA would be along the lateral or Y-axis of the aircraft since there is normally negligible accleration along this axis. The second choice would be along the longitudinal or X-axis of the aircraft where there is minimal or only transient acceleration. The final choice of inertial component orientation will be a compromise based on several similar considerations.

The default orientation of the inertial components appearing in the simulation is illustrated in Figure D-1, where one inertial component of each kind has its input axis parallel to one positive body axis.

D.2 X-inertial Component Orientation

The gyro or accelerometer orientation with respect to the body frame is specified by a rotation matrix, QGBX or QABX, in the program mnemonics, where the default value is:

$$\text{QGBX or QABX} = X\left(\frac{\pi}{2}\right) \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

In general, the nominal orientations for the X-gyro or accelerometer are:

$$\text{QGBX or QABX} = X\left(\frac{k\pi}{2}\right), \; k = 0, 1, 2, 3$$

If one wishes to specify infinitesimal misalignments of the input axis (IA) of the particular X inertial component, I_X, with respect to the body frame, these may be represented by the small angles, β_x or γ_x, about the Y and Z-body axes. The resulting transformation from the body frame to the slightly misaligned X-inertial component frame is given by:

$$\text{QGBX or QABX} = X\left(\frac{k\pi}{2}\right) Y_\varepsilon(\beta_x) \; Z_\varepsilon(\gamma_x)$$

where $Y_\varepsilon(\beta_x)$ and $Z_\varepsilon(\gamma_x)$ denote infinitesimal rotations about the Y and Z body axes respectively and are

$$Y_\varepsilon(\beta_x) \triangleq Y(\beta_x) \Big|_{\beta_x \ll 1} = \begin{bmatrix} 1 & 0 & -\beta_x \\ 0 & 1 & 0 \\ \beta_x & 0 & 1 \end{bmatrix}$$

I_i - input axis of i-gyro or accelerometer

O_i - output axis of i-gyro or accelerometer

*P_i - pendulous axis of i-gyro or accelerometer

*S_i - spin axis of i-gyro or accelerometer

$i = x, y, z$

*Reference Position

$\hat{P}_i = \hat{i}_i \times \hat{o}_i, \hat{S}_i = \hat{i}_i \times \hat{o}_i$

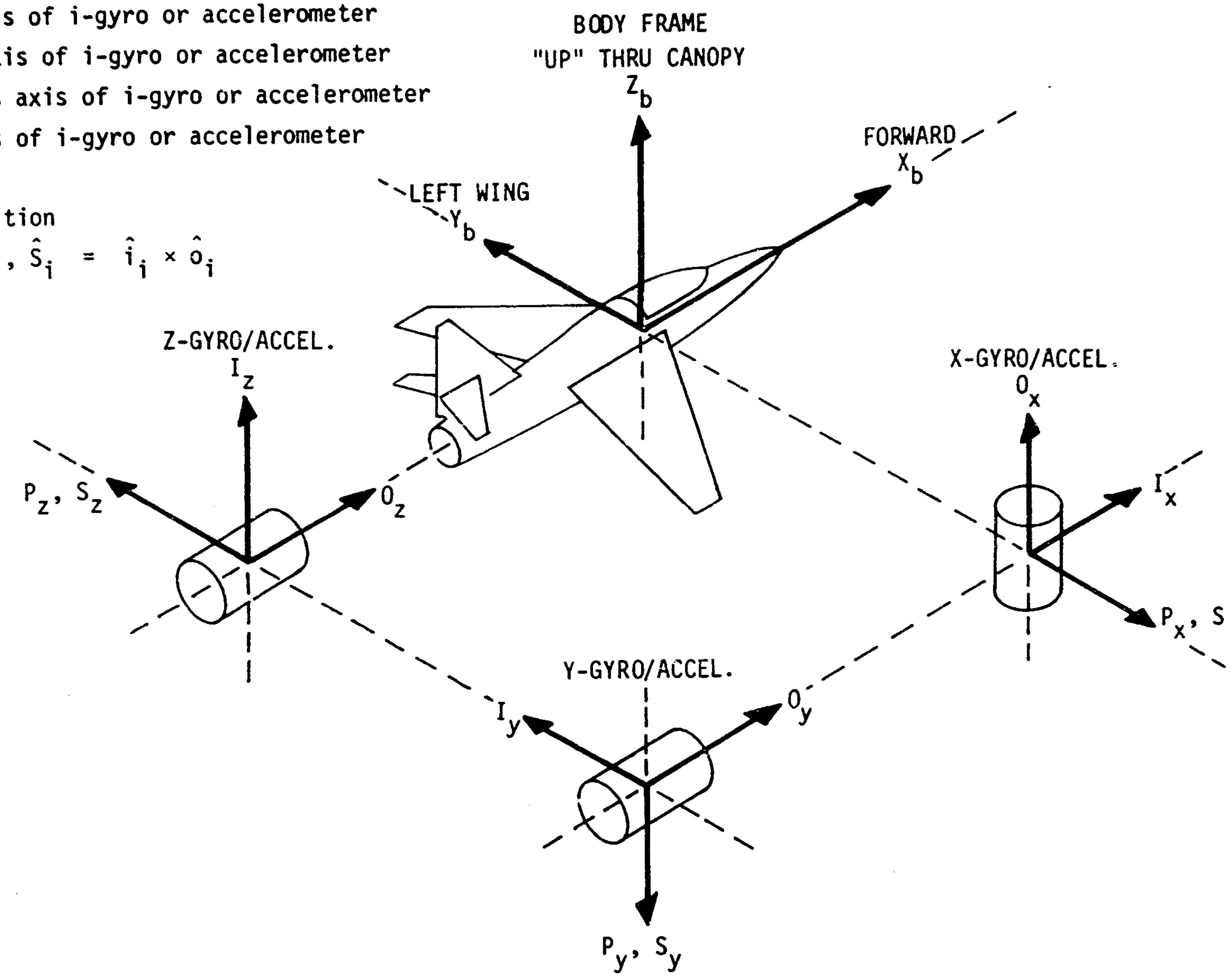


Figure D-1. Body frame and default inertial component orientations.

and

$$Z_\varepsilon(\gamma_x) \triangleq Z(\gamma_x) \Big|_{\gamma_x<<1} = \begin{bmatrix} 1 & \gamma_x & 0 \\ -\gamma_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since β_x and γ_x are small--on the order of a milliradian or less--their products may be neglected for the anticipated applications, and the product of the two may be considered to be multiplicatively commutative, i.e.,

$$Y_\varepsilon(\beta_x)\ Z_\varepsilon(\gamma_y) = Z_\varepsilon(\gamma_y)\ Y_\varepsilon(\beta_x) = \begin{bmatrix} 1 & \gamma_x & -\beta_x \\ -\gamma_x & 1 & 0 \\ \beta_x & 0 & 1 \end{bmatrix}$$

The general transformation from the body frame to the X inertial component frame is:

$$\text{QABX or QGBX} = \left[\begin{array}{c|c|c} 1 & \gamma_x & -\beta_x \\ -\gamma_x c\left(\frac{k\pi}{2}\right) + \beta_x s\left(\frac{k\pi}{2}\right) & c\left(\frac{k\pi}{2}\right) & s\left(\frac{k\pi}{2}\right) \\ +\gamma_x s\left(\frac{k\pi}{2}\right) + \beta_x c\left(\frac{k\pi}{2}\right) & -s\left(\frac{k\pi}{2}\right) & c\left(\frac{k\pi}{2}\right) \end{array}\right]$$

For the default value (k=1) of the X inertial component orientation

$$\underset{(k=1)}{\text{QABX or QGBX}} = \begin{bmatrix} 1 & \gamma_x & -\beta_x \\ \beta_x & 1 & 0 \\ \gamma_x & 0 & 1 \end{bmatrix}$$

of course, β_x and γ_x may be different for the X-gyro than for the X-accelerometer. Misalignment about the input axis, which would be denoted by α_x in this case, is never considered in practice.

D.3 Y-inertial Component Orientation

The Y-gyro or accelerometer orientation with respect to the body frame is specified by a rotation matrix, QGBY or QABY, in the program mnemonics, where the default value is:

$$\text{QGBY or QABY} = X(\pi)\ Z\left(\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

In general, the nominal orientation for the Y-gyro or accelerometer is:

$$\text{QGBY or QABY} = X\left(\frac{k\pi}{2}\right)\ Z\left(\frac{\pi}{2}\right),\ k = 0,\ 1,\ 2,\ 3$$

Infinitesimal misalignments of the IA of the particular Y-inertial component, I_y, with respect to the body frame, may be respresented by small angles, γ_y and α_y, about the Z and X-body axes respectively. The resulting transformation from the body frame to the slightly misaligned Y-inertial component frame is:

$$\text{QGBY or QABY} = X\left(\frac{k\pi}{2}\right) Z\left(\frac{\pi}{2}\right) Z_\varepsilon(\gamma_y)\ X_\varepsilon(\alpha_y)$$

where $Z_\varepsilon(\gamma_y)$ and $X_\varepsilon(\alpha_y)$ denote infinitesimal rotations about the Z and X-body axes respectively and are:

$$Z_\varepsilon(\gamma_y) \triangleq Z(\gamma_y)\Big|_{\gamma_y \ll 1} = \begin{bmatrix} 1 & \gamma_y & 0 \\ -\gamma_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$X_\varepsilon(\alpha_y) \triangleq X(\alpha_y)\Big|_{\alpha_y \ll 1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha_y \\ 0 & -\alpha_y & 1 \end{bmatrix}$$

The general transformation from the body frame to the Y-inertial component frame is:

$$\begin{array}{c} \text{QABY or QGBY} \\ (k = 2) \end{array} = \left[\begin{array}{c|c|c} -\gamma_y & 1 & \alpha_y \\ -C\left(\frac{k\pi}{2}\right) & -\alpha_y\ S\left(\frac{k\pi}{2}\right) - \alpha_y S\left(\frac{k\pi}{2}\right) & S\left(\frac{k\pi}{2}\right) \\ S\left(\frac{k\pi}{2}\right) & \gamma_y\ C\left(\frac{k\pi}{2}\right) - \alpha_y\ C\left(\frac{k}{2}\right) & C\left(\frac{k\pi}{2}\right) \end{array}\right]$$

For the default value (k = 2) of the Y-inertial component orientation

$$\text{QABY or QGBY} = \begin{bmatrix} -\gamma & 1 & \alpha_y \\ 1 & \gamma_y & 0 \\ 0 & \alpha_y & -1 \end{bmatrix}$$

D.4 Z-inertial Component Orientation

Finally, the Z-gyro or accelerometer orientation with respect to the body frame is specified by a rotation matrix, QGBZ or QABZ, in the program mnemonics, where the default value is:

$$\begin{aligned} \text{QGBZ or QABZ} &= X\left(-\frac{\pi}{2}\right) Y\left(-\frac{\pi}{2}\right) \\ &= X\left(\frac{3\pi}{2}\right) Y\left(-\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

In general, the nominal orientations for the Z-gyro or accelerometer are:

$$\text{QGBZ or QABZ} = X\left(\frac{k\pi}{2}\right) Y\left(-\frac{\pi}{2}\right), \ k = 0, 1, 2, 3$$

Infinitesimal misalignments of the IA of the particular Z-inertial component, I_z, with respect to the body frame may be represented by the small angles; α_z and β_z, about the X and Y body axes respectively. The resulting transformation from the body frame to the slightly misaligned Z-inertial component frame is:

$$\text{QGBZ or QABZ} = X\left(\frac{k\pi}{2}\right)Y - \left(\frac{\pi}{2}\right)X_\varepsilon(\alpha_z)\ Y_\varepsilon(\beta_z)$$

where, as before

$$X_\varepsilon(\alpha_z)\ Y_\varepsilon(\beta_z) = Y_\varepsilon(\beta_z)\ X_\varepsilon(\alpha_z) = \begin{bmatrix} 1 & 0 & -\beta_z \\ 0 & 1 & \alpha_z \\ -\beta_z & -\alpha_z & 1 \end{bmatrix}$$

The general transformation from the body frame to the Z-inertial component frame is:

$$\text{QABZ or QGBZ} = \left[\begin{array}{c|c|c} \beta_z & -\alpha_z & 1 \\ -S\left(\frac{k\pi}{2}\right) & C\left(\frac{k\pi}{2}\right) & \alpha_z C\left(\frac{k\pi}{2}\right) + \beta_z S\left(\frac{k\pi}{2}\right) \\ -C\left(\frac{k\pi}{2}\right) & -S\left(\frac{k\pi}{2}\right) & -\alpha_z S\left(\frac{k\pi}{2}\right) + \beta_z C\left(\frac{k\pi}{2}\right) \end{array}\right]$$

For the default value $(k = 3)$ of the Z-inertial component orientation

$$\underset{(k = 3)}{\text{QABZ or QGBZ}} = \begin{bmatrix} \beta_z & -\alpha_z & 1 \\ 1 & 0 & -\beta_z \\ 0 & 1 & \alpha_z \end{bmatrix}$$

D.5 Insertion of Inertial Component Orientation

Each element of each inertial component alignment matrix must be input in both the inertial component module and the corresponding compensation module. It should be reiterated that elements of these matrices are entered row-wise, not columnwise; for instance, if QABZ, using the default nominal orientation and the infinitesimal misalignments, were being input the sequence would be:

$$\beta_z,\ -\alpha_z,\ 1,\ 1,\ 0,\ -\beta_z,\ 0,\ 1,\ \alpha_z.$$

D.6 Component Misalignments and Nonorthogonalities

In each of the gyro and accelerometer compensation modules, there is a provision for an additional misalignment matrix, QMIS. This matrix, which must be entered by the user as a nine element vector, is used to transform the compensated $\underline{\Delta V}^{a}$'s and $\underline{\Delta \theta}^{g}$'s from the accelerometer or gyro input axes to the body frame. (One should note that QBAX in the accelerometer module is the same as QABX in the accelerometer compensation module except for misalignment errors, i.e. one is not supposed to be the inverse of the other). The nine elements of QMIS in the accelerometer compensation module are formed from the elements of the first rows of each of QABX, QABY, and QABZ with the signs of the infinitesimal misalignments reversed, i.e.,

$$\text{QMIS} \triangleq \begin{bmatrix} 1 & -\gamma_x & \beta_x \\ \gamma_y & 1 & -\alpha_y \\ -\beta_z & \alpha_z & 1 \end{bmatrix} = C^{b}_{a_I} ,$$

$$= \begin{bmatrix} QABX_1 & -QABX_2 & -QABX_3 \\ -QABY_1 & QABY_2 & -QABY_3 \\ -QABZ_1 & -QABZ_2 & QABZ_3 \end{bmatrix}$$

Of course, the diagonal elements of QMIS are all unity. QMIS is the "first order inverse" of the nonorthogonal transformation from the body frame to the input axes of the accelerometer triad. Shortening QMIS to Q, the following identity holds:

$$Q = \frac{1}{2}(Q + \tilde{Q}) + \frac{1}{2}(Q - \tilde{Q})$$

$$= \frac{1}{2}\begin{bmatrix} 2 & -\gamma_x + \gamma_y & -\beta_z + \beta_x \\ -\gamma_x + \gamma_y & 2 & -\alpha_y + \alpha_z \\ -\beta_z + \beta_x & -\alpha_y + \alpha_z & 2 \end{bmatrix} \quad \text{Symmetric}$$

$$+ \frac{1}{2}\begin{bmatrix} 0 & -(\gamma_x + \gamma_y) & \beta_z + \beta_x \\ \gamma_x + \gamma_y & 0 & -(\alpha_y + \alpha_z) \\ -(\beta_z + \beta_x) & \alpha_y + \alpha_z & 0 \end{bmatrix} \quad \text{Skew Symmetric}$$

The off diagonal element of the symmetric portion of Q represent the component non orthogonalities. The skew symmetric portion of Q represents the rotational misalignment of the orthogonalized triads.

It should be noted that the α's, β's, and γ's do not necessarily imply the existence of component alignment; only when the values of γ_x and β_x in QABX, for instance, do not coincide with the corresponding values in QBAX do actual (first order) errors in alignment of the X accelerometer propagate into system errors.

The form of the component misalignments used throughout the INSS do imply the existence of independent means of establishing the 'body' frame. This in turn implies the existence of optical cubes, one of which defines the "body" frame fixed in the inertial component cluster.

As it stands, it is evident that the user must exercise great care in setting up a desired set of gyro and accelerometer nonorthogonalities and rotational misalignments.

R-1136

VOLUME I, II, III, IV

DISTRIBUTION LIST

A. Ciccolo
G. Coate
R. Crisp
K. Daly
M. Dare
W. Delaney
W. Denhard
J. DiSorbo
J. Feldman
K. Fertig
J. Fish
J. Gilmore
J. Goode
J. Harper
R. Harris
J. Hursh
P. Kampion
A. Kosol
W. Koenigsberg
R. Leger
J. Lombardo
D. Millard
B. McCoy
P. Motyka
R. Nurse (19)

J. Prohaska
T. Reed
D. Riegsecker
J.P. Ryan
G. Schmidt
J. Sciegienny
L. Schnee
R. Setterlund
T. Thorvaldsen
L.W. Torrey
K. Vincent
P. Volante

TIC (5)

Air Force Avionics Laboratory (30)
Attn: RWA-3 D. Kaiser
Wright-Patterson Air Force Base
Dayton, Ohio 45433

Air Force Avionics Laboratory (5)
Attn: N. Banke RWA-2
Wright-Patterson Air Force Base
Dayton, Ohio 45433